Une architecture Multi-agents pour un apprentissage autonome guidé par les émotions

Jérôme Chapelle

Numéro d'identification :

Académie de Montpellier

Université Montpellier II

— Sciences et Techniques du Languedoc —

Thèse

présentée à l'Université des Sciences et Techniques du Languedoc
pour obtenir le diplôme de DOCTORAT

Spécialité	: Informatique
Formation Doctorale	: *Informatique*
Ecole Doctorale	: *Information, Structures, Systèmes*

Une architecture Multi-agents pour un apprentissage autonome guidé par les émotions

par

Jérôme Chapelle

Soutenue le 11 décembre 2006 devant le Jury composé de :

M. Francois Charpillet, Directeur de Recherches INRIA, INRIA Lorraine, Nancy,.....Rapporteur
M. Jean-Louis Vercher , Directeur de Recherches CNRS, Laboratoire "Mouvement et Perception", Marseille,.....Rapporteur
M. René Zapata , Professeur des Universités, Université Montpellier II, ... Président
M. Stefano Cerri , Professeur des Universités, Université Montpellier II,..... Examinateur
M. Olivier Simonin , Maître de Conférences, détaché CR à l'INRIA Lorraine, Nancy,... Examinateur
M. Jacques Ferber , Professeur, Université Montpellier II, ... Directeur de Thèse

Remerciements

Je remercie respectueusement mon directeur le thèse, le Professeur Jacques Ferber, pour m'avoir transmis sa passion des systèmes multi-agents, pour ses conseils et ses discussions enrichissantes ainsi qu'à la confiance qu'il a placée en moi et grâce à laquelle j'ai été conforté dans mes orientations de recherche.

J'exprime également de sincères remerciements aux membres du jury qui m'ont fait l'honneur de juger ce travail. J'adresse toute ma reconnaissance au Professeur de Robotique René Zapata pour avoir accepté de présider ce jury. Je remercie également le Professeur François Charpillet et le Professeur Louis Vercher qui, par leurs rapports et leurs observations, m'ont permis d'enrichir ce manuscrit.

∞

Ce périple qu'est la thèse n'aurait pu être aussi abouti sans les précieux apports scientifiques de toutes ces personnes que j'ai rencontrées à l'université, en conférence et dans des groupes de travail. Mes remerciements sont donc adressés à toutes ces personnes pour m'avoir donné des pistes intéressantes dans le domaine des sciences du comportement, de l'intelligence artificielle et de la robotique collective. J'ai énormément apprécié ces discussions passionnées et fructueuses dans lesquelles ce travail a pris ses racines et a puisé les ressources qui lui ont permis de progresser.

Par ailleurs l'implication et la motivation que nécessite un tel investissement requièrent également un environnement propice au travail. Aussi, je remercie le Docteur Olivier Simonin pour m'avoir soutenu lors de mon Diplôme d'Etudes Approfondies. Par sa rigueur de travail, il m'a inculqué de bonnes méthodes qui m'ont conduit sur la voie d'une thèse. Merci également au Docteur Fabien Michel qui m'a précédé sur ce terrain et grâce à qui de premières ébauches de qualité de ce mémoire ont pu voir le jour. Mes respects et ma gratitude vont également au Professeur Jocelyne Nanard et au Docteur Tiberiu Stratulat dont l'engagement et la motivation m'ont permis de mener à terme ce travail. Merci à Céline Fiot, au Docteur Abdelkader Gouaich et au Docteur Lylia Abrouk pour leurs relectures approfondies et leurs conseils éclairés qui ont considérablement contribué à améliorer la qualité de ce document. Je remercie avec tout autant de respect ceux à côté de qui j'ai travaillé. Merci à José Baez et à Grégory Beurier pour ces longues discussions philosophiques et leurs soutiens psychologiques qui m'ont été d'un grand secours dans les moments difficiles. Merci aux Docteurs Philippe

Lucidarme, Frédéric Souchon, Didier Schwab, Simon Jaillet, Clément Jonquet. J'adresse mes remerciements et mes encouragements à Mehdi Yousfi Monod, John Tranier, Saber Mansour, Benoit Darties, Chedy Raïssi, Marc Plantevit. Je vous souhaite de connaître l'immense plaisir qu'apporte la finalisation de ce travail.

Je pense également à tous ceux qui m'ont entouré et supporté durant ces années, et plus particulièrement pendant la phase finale durant laquelle leur patience et leur soutien ont été mis à rude épreuve. Merci donc à Patrice Pacaud, Nicolas Mooret, Géraldine Blanc, Frédéric Pons et Clément Durand Daubin pour les agréables moments que nous avons passés ensemble.

Cette aventure n'aurait probablement pas vu le jour sans tous ces enseignants de qualité qui m'ont donné l'envie d'apprendre, de progresser, et m'ont ainsi permis de suivre cette voie.

Enfin, je tiens à remercier chaleureusement mes parents qui m'ont soutenu durant toute ma scolarité dans la voie que j'ai choisie. Je remercie respectueusement mon père pour m'avoir montré les bénéfices à être exigeant et rigoureux dans son travail. Je remercie affectueusement ma mère qui m'a appris à persévérer et à relever la tête même lorsque lorsque le fleuve de la vie rencontre de grands obstacles. Je remercie tendrement ma soeur pour m'avoir encouragé et permis de m'évader un peu dans les moments critiques de la phase finale de rédaction.

Je remercie également la personne qui lit ces lignes pour l'intérêt qu'elle porte à ce document. J'invite également les personnes que j'aurais oubliées de citer, à compléter la ligne ci-dessous afin d'apparaître également dans les présents remerciements (rayez au besoin les mentions inutiles).

Je remercie ..... pour ses nombreux conseils éclairés et pour son soutien amical dans les moments difficiles.

« *Celui qui déplace la montagne,*
c'est celui qui commence par enlever les petites pierres. »
Confucius

Sommaire

Chapitre 1

Introduction

1.1 Cadre général

Cette thèse se situe à la rencontre de deux domaines de recherches. Elle s'inscrit, en effet, dans le domaine de l'Intelligence Artificielle Distribuée (IAD), d'une part, et plus particulièrement dans le domaine des systèmes multi-agents, dans le domaine de la robotique d'autre part. Notre travail concerne ainsi les problèmes d'*apprentissage* et d'*adaptation* rencontrés lors de la réalisation de structures de contrôle destinées à des robots réels.

Grâce aux progrès de miniaturisation, les robots mobiles disposent de moyens de plus en plus précis et variés d'interagir avec l'environnement . En contrepartie, les structures de contrôle sont de plus en plus complexes à développer et deviennent, de fait, spécifiques à des architectures matérielles données. Il apparaît donc utile et même nécessaire de définir une architecture qui puisse générer une structure de contrôle évolutive et adaptative afin que le robot soit capable d'apprendre des tâches de plus en plus complexes et non connues à priori. En outre, la grande diversité de capteurs et d'effecteurs disponibles pour la conception de robots ainsi que la possibilité d'en ajouter de nouveaux à un robot existant, soulèvent des problèmes dus à l'hétérogénéité et à l'évolution du nombre de composants qui constituent la structure physique du robot. La réalisation d'une telle structure, assurant l'autonomie d'un robot disposant de composants hétérogènes, requiert des moyens d'apprentissage et d'adaptabilité. Ces problèmes ont ainsi suscité de nombreux travaux en robotique et en informatique ces dernières années. En effet l'apprentissage automatique est un sous domaine majeur de l'intelligence artificielle qui propose de nombreuses techniques telles que les réseaux de neurones, les arbres de décisions, les Processus Décisionnels de Markov, ou encore les algorithmes et la programmation génétique.

1.2 Démarche adoptée

Observer les êtres vivants

De tout temps l'homme s'est inspiré de la nature pour concevoir et développer des objets manufacturés. Beaucoup d'inventeurs ont su observer les créations de la nature pour réaliser des objets, des machines ou pour élaborer des théories encore exploitées aujourd'hui. Parmi eux, Léonard de Vinci esquisse, en 1485, la "machine volante" (figure 1.1), issue de ses études sur le *Vol des oiseaux*, et établit certains principes repris plus tard par l'aviation.

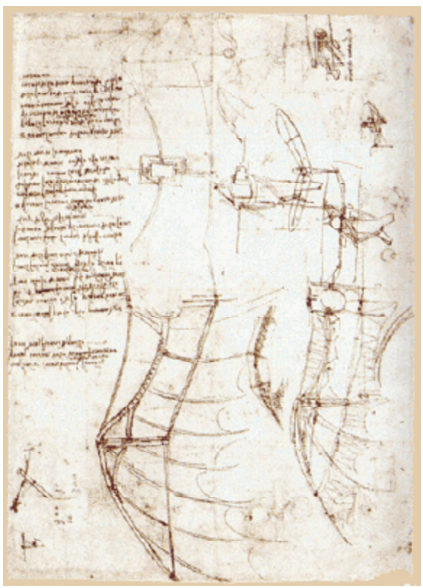

FIG. 1.1 – Etude d'une grande aile aux extrémités manœuvrables, Léonard de Vinci, v.1485.

Grâce à cette démarche, de grandes avancées ont été possibles dans de nombreux domaines comme en médecine pour l'élaboration de médicaments ou en électronique pour la conception du sonar, inspiré par le système d'écholocation de la chauve souris. En informatique, et plus particulièrement en Intelligence Artificielle Distribuée, certains modèles de systèmes multi-agents sont fortement inspirés de l'observation des colonies de fourmis [Ferber, 1999] [Drogoul, 1993]. Dans notre cas nous allons adopter une approche similaire et prendre pour inspiration le cerveau des êtres vivants, car ceux-ci possèdent les propriétés d'adaptation et d'évolution qui permettent à l'entité apprenante de survivre dans son environnement et d'apprendre de nouvelles tâches de plus en plus complexes. Ainsi seront construits des mécanismes que nous considérons comme importants pour les processus d'apprentissage et d'accumulation de savoir-faire au sein d'une entité autonome évoluant et se déplaçant dans un environnement dynamique.

Modéliser une architecture de contrôle

L'idée de s'inspirer des systèmes nerveux des espèces vivantes dans le but de doter les machines d'intelligence artificielle remonte maintenant à plusieurs décennies. En effet, les concepts de neurone artificiel et de réseau de neurones artificiels ont été introduits peu avant les années 60 et ont depuis donné lieu à de nombreux travaux et à de nombreuses applications. Ils ont montré leur supériorité à résoudre des problèmes jusque-là réservés aux espèces animales comme la reconnaissance de formes ou de sons. Mais leur utilisation exclusive dans la réalisation de structures de contrôle permettant de construire de nouveaux comportements pose des problèmes liés à une structuration assez rigide.

Plus récemment, à la fin des années 80, des recherches dans le domaine des neurosciences ont mis en évidence l'existence de regroupements de neurones dupliqués à grande échelle au sein du cortex (la partie supérieure du cerveau). Cette modélisation corticale permet de considérer ces assemblages de neurones comme des unités de calculs ayant des capacités fonctionnelles plus importantes que celles d'un simple neurone. Le fait de modéliser un système nerveux artificiel avec une granularité moins fine qu'en modélisant chaque neurone apparaît être alors un bon moyen pour réaliser des architectures de contrôle apprenantes plus souples et plus proches des systèmes nerveux des espèces animales.

Mettre en relation un grand nombre d'entités apprenantes et autonomes s'inscrit également dans le courant de pensée ouvert par Marvin Lee Minsky au milieu des années 80. Dans son livre *La Société de l'esprit* [Minsky, 1986] il présente l'esprit comme une architecture d'entités autonomes qui, de par leurs interactions entre elles, permettent à l'intelligence d'émerger. Réaliser de tels systèmes demande un effort de modélisation pour rendre efficace leur mise en place sur les architectures matérielles dont nous disposons. En effet les calculateurs que nous utilisons pour animer nos entités artificielles sont structurellement différents des systèmes nerveux dont on souhaite s'inspirer. De plus les robots que l'on souhaite équiper d'une telle architecture de contrôle diffèrent entre eux de par leur capacités physiques à agir sur l'environnement. Il est donc important de concevoir des modèles capables de s'adapter aux architectures matérielles sur lesquels il seront déployés. Nous allons donc devoir nous positionner entre une modélisation fidèle d'un système nerveux biologique et des modélisations mettant en œuvre des modèles mathématiques et logiques. Pour cela nous étudierons plusieurs architectures de contrôle qui ont déjà fait leurs preuves en intelligence artificielle.

Apprendre et s'adapter

L'émergence d'une structure de contrôle permettant la réalisation de nouvelles tâches et l'adaptation de cette structure aux évolutions de l'environnement et à la physique

du robot nécessitent l'utilisation de méthodes d'apprentissage. Les travaux antérieurs qui traitent d'apprentissage sont souvent appliqués à des environnements simulés et discrétisés : les mondes grille [Maçãs *et al.* , 2001] [Dutech *et al.* , 2001]. Ces travaux sont souvent éloignés des problèmes rencontrés en robotique mobile (en particulier pour ce qui est de la perception de l'environnement). Une autre approche classique pour réaliser un apprentissage en robotique collective consiste à utiliser des communications complexes et riches entre les agents (comme dans [Parker, 1998] et [Shibata *et al.* , 1996]), mais de telles solutions voient généralement leur utilité limitée à des problèmes spécifiques. De plus, dans beaucoup de travaux, le processus d'apprentissage est supervisé par un observateur externe qui donne à l'entité apprenante une récompense pour motiver le processus d'apprentissage.

Pour pouvoir se passer d'un superviseur, il est important de bien définir les motivations qui vont guider le robot dans son processus d'apprentissage. Pour garantir l'autonomie de processus, et celle du robot, il faut que ces motivations permettent au robot de percevoir les situations critiques pour lui, ou pour la tâche qu'il doit accomplir. De nombreux travaux, [Matarić, 1994b] [Gadanho & Hallam, 2001a] [Gadanho & Hallam, 2001b] [Foliot & Michel, 1998], montrent l'intérêt d'utiliser des émotions, qui sont définies comme des motivations, et qui sont très similaires aux besoins primaires auxquels les espèces vivantes sont assujetties, comme la faim, le besoin de jouer, d'être en société ou de se reproduire.

Nous allons donc proposer un modèle qui permette de générer des structures de contrôle adaptatives et évolutives motivées par des besoins liés à la physique du robot et aux tâches pour lesquelles celui-ci à été construit. Pour permettre cela, nous allons proposer une architecture multi-agents dont le principe de fonctionnement est fondé sur le principe du modèle d'interaction cérébro - cérébelleux, [Ito, 1984], et qui utilise des émotions pour guider l'apprentissage.

1.3 Plan de thèse

Les travaux présentés dans ce manuscrit sont organisés de la manière suivante :

Une première partie présente un état de l'art des modèles et des solutions existantes. Tout d'abord, dans le chapitre 2, sont présentées les inspirations prises par observation du vivant : les réseaux de neurones et les assemblées neuronales. Puis dans le chapitre 3, nous présentons différentes méthodes de modélisation permettant de réaliser des architectures de contrôle. Nous exposons alors plusieurs architectures de contrôle existantes utilisées dans la mise en œuvre de robots, ce qui nous permet de justifier les choix de modélisation que nous avons faits.

Dans une deuxième partie, nous présentons notre modèle ainsi que les processus d'apprentissage permettant l'émergence et l'adaptation d'une architecture de contrôle distribuée. Le chapitre 4 décrit le système multi-agents et les raisons qui nous ont amené à ce choix. Puis le chapitre 5 présente les techniques utilisées pour permettre l'émergence d'une structure de contrôle au sein du système multi-agents. Dans cette section seront introduites les deux phases du processus d'apprentissage : le *babbling learning* et le *satisfying learning*.

La troisième et dernière partie présente une implémentation de notre modèle et une analyse de ses capacités. Dans le chapitre 6, le modèle est évalué ainsi que ses applications potentielles aux problèmes classiques de robotique mobile. Un exemple d'application de cette architecture sur un cas concret est également présenté dans ce chapitre. Enfin, le chapitre 7 dresse un bilan des capacités de notre architecture et présente les diverses perspectives ouvertes.

Chapitre 2

Les inspirations du vivant : les neurosciences

« La raison pour laquelle les ordinateurs sont incapables de traiter des problèmes de sémantique devient évidente. Leur mise en œuvre n'est pas la bonne ; elle ne mène pas à la conscience. » Gerald Edelman [Edelman, 1992]

L'intelligence artificielle est une des disciplines informatiques qui fait beaucoup parler d'elle, y compris dans les milieux non-scientifiques. Cela peut s'expliquer par le fait que cette discipline est relativement récente à l'échelle de l'histoire. En effet, jusqu'à ce siècle, les outils et objets manufacturés par l'homme lui permettaient de prolonger son corps en améliorant ou en ajoutant certaines capacités d'action, puis de perception. Pour illustrer ce propos on peut considérer la roue comme un outil nous permettant d'améliorer les capacités de déplacement offertes par nos jambes. De la même manière, une longue vue nous permet, comme son nom l'indique, d'améliorer notre capacité à voir. Les travaux qui traitent d'intelligence artificielle se posent l'objectif de prolonger l'esprit de l'être humain en déportant, sur l'outil réalisé, une partie de l'intelligence humaine.

L'intelligence artificielle, discipline à laquelle beaucoup de travaux revendiquent leur appartenance, s'est donc initialement développée autour du traitement d'informations symboliques, de raisonnements logiques, structurés et explicites. En effet, les machines ont montré leur supériorité sur les humains pour traiter de grands nombres d'inférences, et de grandes quantités de données. Les systèmes experts, qui opèrent sur une base de règles avec un moteur d'inférence ou encore les algorithmes de recherche dans des structures symboliques, arbres et graphes, permettent de remplacer l'intelligence humaine lorsque le problème à traiter nécessite un raisonnement logique. Mais qu'en est-il lorsque l'on veut obtenir d'un outil qu'il apprenne par lui-même, réagisse « correctement » à des situations imprévues et adapte dynamiquement son comportement à son environnement ou à de nouveaux besoins ? Si l'on suit les idées soutenues par le créationnisme,

l'intelligence serait un don d'une entité « supérieure » responsable de notre présence sur terre. N'ayant à ce jour que peu d'informations vérifiables à ce sujet et préférant les approches scientifiques, nous choisissons de considérer la théorie de l'évolution selon Darwin. En effet, les systèmes vivants que l'on peut observer dans la nature résolvent plutôt bien les problèmes d'adaptation, ce qui confère aux espèces les mieux adaptées la possibilité pour elles de se pérenniser et à leurs représentants la capacité de survivre dans notre environnement qu'est la planète terre.

Dans cette section, nous allons donc nous intéresser à ce qui se fait dans la nature en terme d'intelligence comportementale. Cependant, pour obtenir une intelligence artificielle, on peut se demander s'il est pertinent de vouloir reproduire avec exactitude tous les acteurs et les processus impliqués dans l'intelligence biologique. Notre avis sur la question est qu'il faut s'approcher le plus possible des modèles biologiques, mais sans perdre de vue que ces modèles peuvent ne pas être adaptés aux systèmes artificiels existants, et en particulier aux plates-formes informatiques, sur lesquels ils sont déployés. C'est pourquoi, dans cette section, nous nous intéressons aux neurosciences et allons détailler les modèles et les acteurs dont nous avons choisi de nous inspirer pour la conception de notre modèle d'apprentissage et la réalisation de l'architecture correspondante.

2.1 Les réseaux de neurones

2.1.1 Introduction et historique

Les systèmes nerveux qui composent les cerveaux des espèces vivantes ont motivé beaucoup de travaux et de recherches. C'est en 1943 que deux bio-physiciens, Warren S. Mac Cullogh et Walter Pitts introduisent le premier modèle de neurone biologique [Cullogh & Pitts, 1943]. Le neurone, aussi appelé neurone à seuil, y est modélisé comme une fonction, au sens mathématique du terme, c'est-à-dire une association entre des valeurs d'entrée et une valeur de sortie. Six années plus tard, Donald Hebb s'intéresse à des modèles massivement parallèles et propose de nombreuses règles de mise-à-jour des poids [Hebb, 1949]. Plus tard, le psychologue Franz Rosenblatt s'inspire du système visuel et propose le premier perceptron [Rosenblatt, 1958]. Ce réseau permet d'apprendre à différencier des formes simples et à calculer certaines fonctions logiques. Plus tard, des limitations de ce réseau seront démontrées par Minsky et Papert [Minsky & Papert, 1969], et l'engouement pour les réseaux de neurones ne reprendra que pendant les années 80. C'est alors que Hopfield démontre l'intérêt des réseaux entièrement connectés [Hopfield, 1982] et que Rumelhart conçoit le mécanisme de rétropropagation. Ce mécanisme permet de propager l'erreur vers les couches cachées des réseaux multicouches de type perceptron [James L. McClelland & the PDP Research Group, 1986]. Depuis ces travaux, les

réseaux de neurones ont été utilisés et appliqués en informatique [Kohonen, 1990], en intelligence artificielle [Revel, 1997] et en robotique [Lucidarme, 2003]. Les réseaux de neurones ont ainsi montré leur utilité pour résoudre des problèmes jusque-là réservés au monde du vivant comme la reconnaissance de forme, de symboles, ou de la voix humaine. Ces problèmes peuvent être en effet traités grâce à ce genre d'architecture parallèle qui se prête particulièrement bien aux problèmes d'approximation distribuée et d'apprentissage.

2.1.2 Modèle général d'un neurone

Afin de présenter ce qu'est un réseau de neurones artificiel, il convient de définir le modèle général d'un neurone artificiel. Un neurone artificiel j, comme celui présenté sur la figure 2.1, est constitué des éléments suivants :

- n liaisons synaptiques, qui représentent les stimulus en entrée, notés x_i, et pondérés par les poids w_{ij} ($n \geq 1$),
- une fonction qui somme les entrées pondérées, notée Σ,
- une fonction de transfert $\sigma()$ qui détermine,
- la sortie S du neurone

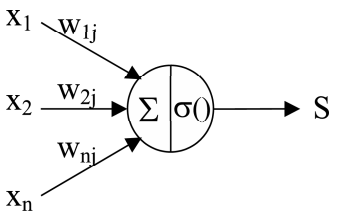

FIG. 2.1 – Schéma général d'un neurone artificiel.

La sortie S peut donc être calculée à partir des n entrées en utilisant la fonction mathématique suivante :

$$S = \sigma(\sum_{j=1}^{n} x_j \cdot w_{ij})$$

2.1.3 Principe de fonctionnement d'un réseau de neurones

Pour former un réseau de neurones, les neurones artificiels sont reliés entre eux par leurs liaisons synaptiques. Dès lors, il est possible de représenter un réseau de neurones comme un système d'équations mathématiques. Comme nous l'avons vu précédemment, chacune des liaisons synaptiques est pondérée. Le comportement du réseau de neurones dépend donc des poids synaptiques. Le processus d'apprentissage consiste donc à modifier ces poids pour obtenir du réseau le comportement souhaité.

Dans un *réseau sans couche cachée*, il y a autant de neurones que de sorties. Chaque neurone est directement relié à toutes les entrées par ses liaisons synaptiques comme sur la figure 2.2. L'apprentissage est le processus qui consiste à calculer l'erreur entre la sortie produite et la sortie désirée. Puis à ajuster les poids synaptiques correspondants afin de diminuer l'erreur. La mise-à-jour des poids est faite en utilisant un coefficient d'apprentissage. Plus ce dernier est faible, plus l'apprentissage sera long et stable. Ce concept de plasticité neuronale, matérialisé par ce coefficient d'apprentissage, est d'ailleurs aussi considéré dans les méthodes d'apprentissage par renforcement que nous verrons en section 4.1.2. Cependant les réseaux de neurones sans couche cachée ne permettent de modéliser que des fonctions linéaires, ce qui limite leur utilisation.

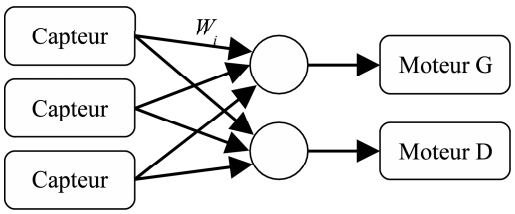

FIG. 2.2 – Exemple d'utilisation d'un réseau de neurone dans une architecture réactive.

En rajoutant une ou plusieurs couches intermédiaires supplémentaires on obtient ce que l'on appelle un *réseau avec couche cachée*, comme celui présenté sur la figure 2.3. Cet ajout de couche(s) permet donc au réseau de distinguer plusieurs perceptions intéressantes, et accroit le nombre de fonctions modélisables. Dans ce type de réseau, la méthode la plus connue est basée sur la rétropropagation de gradients. Cette méthode consiste à propager l'erreur entre la sortie obtenue et la sortie désirée à travers le réseau de neurones afin de modifier tous les poids, même ceux des couches cachées.

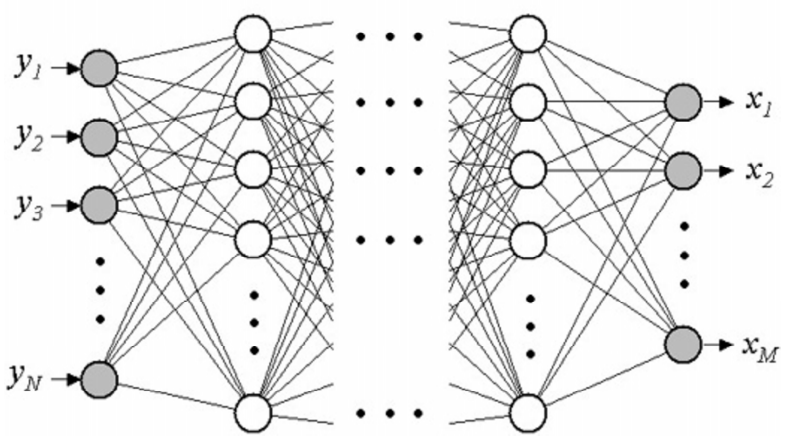

FIG. 2.3 – Exemple d'un réseau à plusieurs couches (MLP : Multi Layered Perceptron).

2.1.4 Discussion

Bien que les réseaux de neurones artificiels classiques permettent d'obtenir de bons résultats sur certains problèmes d'apprentissage, ils semblent trop rigides pour apprendre et mémoriser comme souligné dans [Beaugé & Alexandre, 1995], surtout quand on les compare aux performances humaines. D'une part, parce que les architectures matérielles dont nous disposons sont fortement limités au regard des systèmes nerveux biologiques que l'on simule et qui sont des systèmes massivement parallèles. D'autre part, une difficulté à utiliser des réseaux de neurones réside dans le choix du bon modèle de réseau et de sa complexité structurelle pour un problème donné. Si le modèle a trop de paramètres, on rencontre un problème d'overfitting. Ceci signifie que le réseau de neurones va avoir une erreur de généralisation plus importante que l'erreur empirique, ce qui l'amène à apprendre plus que ce qui n'est nécessaire (du bruit, par exemple).

2.2 Le choix du cortex comme modèle de traitement de l'information

Le neurone formel défini par Rosenblatt [Rosenblatt, 1958] est certes un modèle de neurone, mais la validité biologique des réseaux à couche est discutée. En outre, le perceptron multicouche et la règle d'apprentissage associée représentent surtout des techniques ad hoc à la résolution de problèmes de classification supervisée ce qui limite leur utilisation pour les problèmes qui nous préoccupent.

Pour remédier à ce manque de flexibilité des réseaux de neurones, d'autres travaux [Burnod, 1989] [Frolov *et al.* , 2002] [Frankowska *et al.* , 2003] [Frezza-Buet, 1999] suggèrent de prendre comme point de départ le cerveau des êtres vivants, mais avec une granularité moins fine que le neurone. En effet, le cortex est considéré par Edelman [Edelman, 1992] comme une unité de traitement fondamentalement plus puissante que celles mises en œuvre actuellement sur les ordinateurs. L'équipe CORTEX du Loria [Cortex, n.d.], placée sous la responsabilité scientifique de Frédéric Alexandre, utilise un modèle cortical basé sur les concepts d'assemblée neuronale et de colonnes corticales. Nous allons voir ici comment se présente l'organisation de cette structure nerveuse et les particularités qui nous intéressent dans nos travaux.

2.2.1 Présentation du cortex

Cortex en latin signifie « écorce » ce qui rend bien compte de la structure du cortex cérébral. Le cortex se présente comme une fine couche, d'une épaisseur d'environ deux millimètres, qui entoure la partie supérieure du cerveau. Cette structure essentiel-

lement bidimensionnelle a, chez l'homme, une surface de l'ordre du quart de mètre carré ($0{,}25m^2$). Elle est repliée sur elle-même, ce qui donne au cerveau cette apparence d'un gros torchon blanc roulé en boule (figure 2.4). Les éléments qui constituent le cortex sont connectés localement entre eux, comme le sont les nœuds du maillage d'une surface. Des modèles, comme les cartes auto-organisatrices (**S**elf **O**rganizing **M**aps) de Kohonen [Kohonen, 1990] s'inspirent des propriétés de ce voisinage bidimensionnel pour définir des réseaux de neurones connectés latéralement, par opposition aux réseaux à couches usuels.

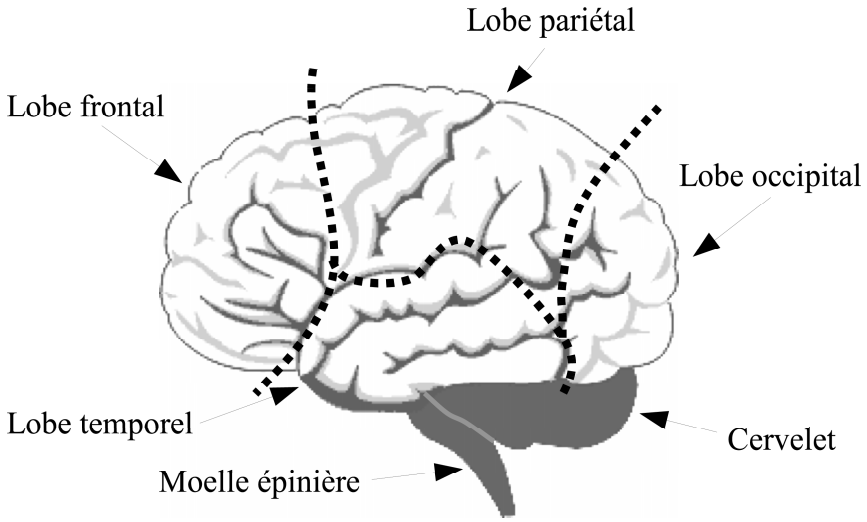

FIG. 2.4 – Vue d'ensemble des différentes régions du cortex (en blanc) et du cervelet (en sombre).

Dans les sous-sections qui suivent, nous allons présenter par granularité croissante une manière de considérer le découpage du cortex en unités élémentaires de calcul : *les colonnes corticales* présentées en sous-section 2.2.2. Puis nous verrons que ces unités élémentaires peuvent êtres regroupées en entités fonctionnelles de plus haut niveau : *les aires corticales*, présentées en sous-section 2.2.3. Enfin nous terminerons cette section par une discussion (sous-section 2.2.4) expliquant de quelles fonctionnalités nous souhaitons nous inspirer pour réaliser une structure de contrôle qui possède des capacités d'évolution et d'adaptation.

2.2.2 Les colonnes corticales

Le cortex est une structure extrêmement régulière au sein de laquelle sont dupliqués des groupes de neurones. Ces groupements, qui peuvent se recouvrir partiellement, forment la surface corticale, voir illustration figure 2.5. Cette surface peut alors être comparée à une structure formée par la juxtaposition de petites colonnes dont l'axe

serait perpendiculaire à la surface en question. On appelle ces petits modules fonctionnels des *colonnes corticales* [Burnod, 1989] [Alexandre, 1990] [Alexandre, 1997]. Dans le cerveau chaque colonne est constituée d'une centaine de neurones environ, et mesure environ $30\mu m$. Les colonnes possèdent une connectivité de proximité avec les colonnes voisines, ainsi qu'une connectivité à grande distance avec d'autres parties du cortex, même si la connectivité locale est prépondérante.

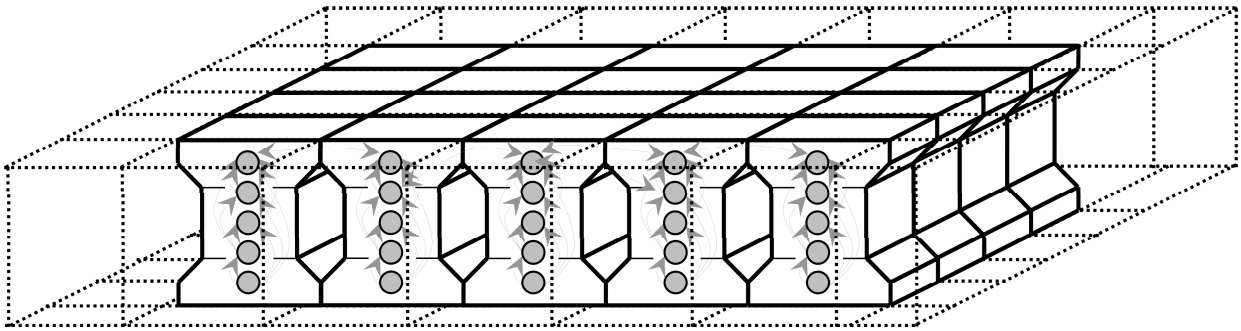

FIG. 2.5 – Représentation de la surface corticale comme juxtaposition de colonnes.

On peut décrire schématiquement la structure d'une colonne corticale en découpant le cortex en six couches parallèles à la surface de celui-ci. Nous regroupons certaines de ces couches, et présentons ici trois types de couches regroupées par leurs caractéristiques :

- Les trois couches supérieures assurent la communication avec des colonnes voisines (voir figure 2.6). Entre colonnes contiguës, ces trois couches se recouvrent partiellement.
- La couche centrale reçoit les excitations perceptives du thalamus[1].
- Les deux dernières couches envoient les informations vers des structures cérébrales autres que le cortex.

Lorsque la couche centrale d'une colonne reçoit des excitations du thalamus, cela ne suffit pas à l'exciter pleinement. De la même manière, lorsque la colonne est soumise à une excitation cortico-corticale, cela la sensibilise, mais ne suffit pas à l'exciter. Pour simplifier, on peut considérer que chaque colonne est associée à un événement perceptif. Lorsque cet événement est nécessaire au comportement, l'apprentissage au sein du cortex fait que la colonne est sensibilisée par ses voisines. L'événement perceptif demandé par le cortex se propage vers les colonnes motrices de sorte que l'événement attendu se produise.

[1]Thalamus : du Grec *Thalamos* = chambre nuptiale. Partie du cerveau située au centre de celui-ci qui a pour rôle de relayer les impulsions nerveuses qui transportent les informations sensorielles. Le thalamus reçoit ces entrées perceptives ainsi que des flux provenant d'autres parties du cerveau, et détermine parmi ces signaux ceux qui doivent être transmis au cortex. Il joue donc, pour le cortex, un rôle d'excitateur et d'inhibiteur.

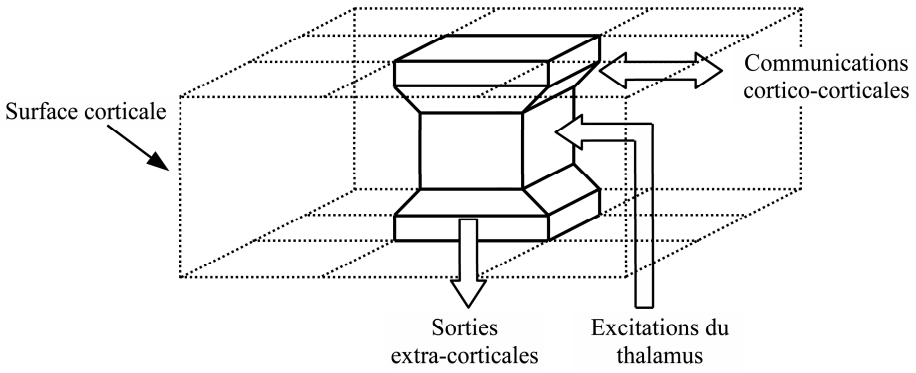

FIG. 2.6 – Interactions d'une colonne corticale.

Dans notre approche, nous considérerons deux aspects importants de ce modèle. Tout d'abord, nous voyons les colonnes corticales comme les plus petites unités de calcul du cortex et ne descendrons pas jusqu'à la simulation des centaines de neurones qui composent chacune de ces colonnes. En effet, les fonctionnalités offertes par plusieurs neurones peuvent êtres obtenues en utilisant une seule entité apprenante. Chaque colonne réalise des opérations élémentaires très spécialisées, mais d'un niveau fonctionnel plus élevé que le simple neurone formel. Ensuite, l'appel peut être vu comme un but perceptif à résoudre. La propagation de cet appel peut être vue comme la recherche de buts intermédiaires, plus facilement atteignables et transmis à d'autres colonnes du cortex.

2.2.3 Les aires corticales

Dans la section précédente, nous avons considéré les colonnes corticales comme les unités élémentaires de traitement de l'information de par la redondance des structures de neurones qui les forment. Or ces colonnes se retrouvent également dupliquées : plusieurs colonnes peuvent recevoir les mêmes entrées, comme le font les neurones des cartes de Kohonen [Kohonen, 1990]. On peut alors choisir de regrouper ces colonnes en *aires corticales*. Chaque aire corticale est typée par la nature des informations qu'elle traite. On distingue alors quatre catégories d'aires :

Les *aires sensorielles*

Ce sont les aires qui sont connectées aux perceptions. L'organisation du cortex sensoriel correspond à la topographie des différents moyens de perception. En effet, les flux d'informations sensorielles sont distribués sur la surface corticale de manière à ce que les relations de voisinage entre les capteurs soient préservées. On parle alors, par exemple, de rétinotopie pour désigner la cartographie des aires visuelles.

Les *aires motrices*

Les aires motrices qui sont câblées aux effecteurs réalisent les sorties vers le monde extérieur, et permettent ainsi l'expression du comportement. Comme pour les aires sensorielles, la topologie des aires motrices est proche du schéma corporel.

Les *aires associatives*

Ces aires connectent les aires sensorielles et motrices entre elles. Elles permettent de faire la relation entre le schéma corporel et les informations sensorielles. Ce sont ces aires qui permettent de réaliser la coordination œil - main à partir des informations visuelles, ou encore de localiser une source sonore à partir des informations auditives. Les aires associatives ont donc pour rôle de créer, à partir d'informations sensorielles ou motrices élémentaires, des représentations de plus haut niveau et plus structurées (Figure 2.7).

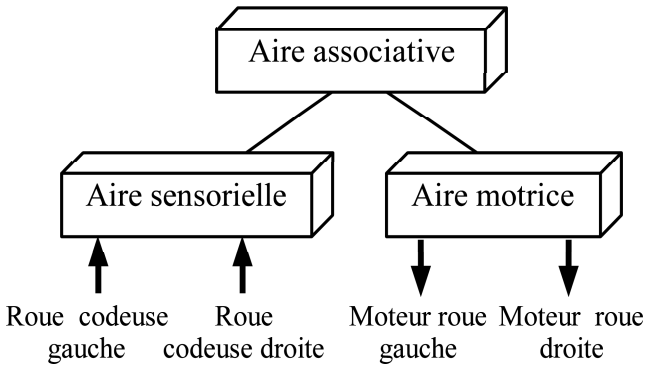

FIG. 2.7 – Illustration de la construction d'une aire associative.

Les *aires frontales*

Ce sont ces aires qui contrôlent et coordonnent le fonctionnement des aires associatives. Elles élaborent l'organisation temporelle du comportement grâce à des rythmes biologiques internes provenant de structures extra-corticales et déterminent ainsi l'ordonnancement des comportements. De manière générale, on peut les considérer comme les aires responsables des comportements de haut niveau comme l'anticipation, la planification ou la résolution de problèmes.

2.2.4 Discussion et exemple d'application

Pour notre approche, nous garderons de ce modèle l'idée que plusieurs unités élémentaires de traitement de l'information (les colonnes) forment un groupe (une aire) au

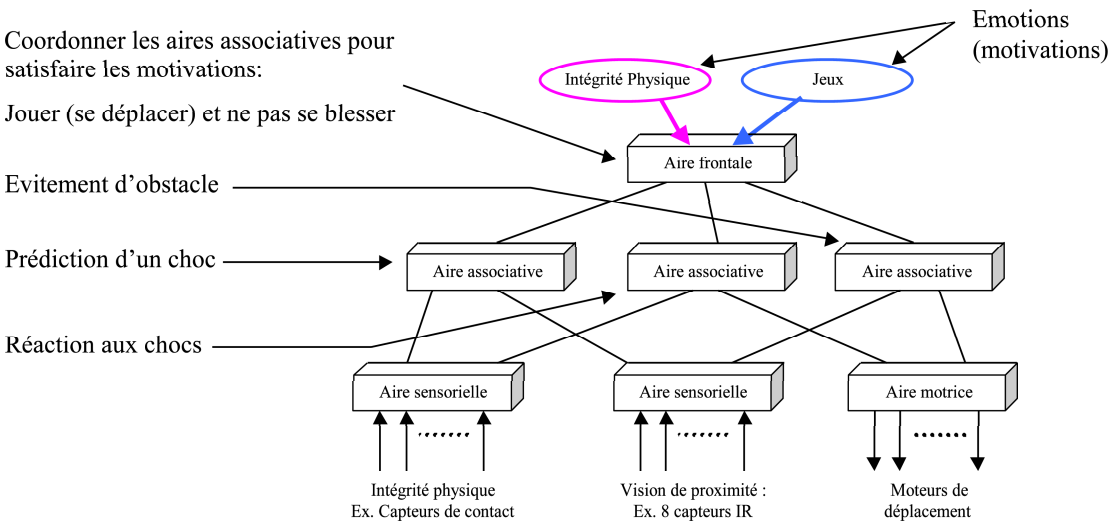

FIG. 2.8 – Exemple de construction des aires corticales mises en jeux sur un robot mobile dans le cadre d'un exercice d'évitement d'obstacles. Pour guider l'apprentissage et obtenir le comportement voulu, la structure de contrôle est soumise aux motivations associées au jeu et à l'intégrité physique du robot.

sein duquel elles interagissent et collaborent pour fournir la propriété qu'a chaque aire corticale à traiter un type d'information. La figure 2.8 montre l'assemblage d'aires qui pourrait être utilisé pour réaliser la structure de contrôle d'un petit robot mobile devant se déplacer dans un environnement tout en évitant les obstacles. L'aire associative qui se situe à gauche de la figure est connectée aux capteurs de contact ainsi qu'aux capteurs infrarouges. Cette aire sera donc en mesure d'apprendre qu'à partir d'une certaine distance mesurée avec les capteurs infrarouges, le choc devient inévitable. Elle disposera donc de l'information permettant d'anticiper le choc. L'aire associative qui est liée aux capteurs de contact et aux moteurs (au centre de la figure), doit quant à elle apprendre à avoir la bonne réaction aux chocs : par exemple en faisant se déplacer le robot dans la direction contraire au contact. Enfin, l'aire qui est connectée à la vision de proximité et aux moteurs doit apprendre à faire varier la vitesse et la trajectoire du robot afin de maximiser la vitesse de déplacement tout en évitant les obstacles.

Cette structure peut néanmoins présenter des défauts, ou manquer d'un comportement intelligent auquel le concepteur n'aurait pas pensé. En effet on ne prend pas en compte le cas où un acteur externe heurterait le robot. Dans ce cas, le robot doit éviter de réagir au choc en prenant une trajectoire sur laquelle se trouve un obstacle. On voit qu'il est donc nécessaire avant de réagir à un choc de prendre en compte la prédiction d'un autre choc potentiel. Pour remédier à cela, il faut que cette structure de contrôle puisse évoluer en fonction des besoins. Dans ce cas précis, il faut permettre l'ajout d'une aire associative supplémentaire. Cette aire associative serait connectée aux deux aires

associatives qui prédisent et réagissent aux chocs, et permettrait donc de mieux réagir aux chocs.

Chapitre 3

Architectures de contrôle existantes

DANS la section précédente, nous avons exprimé notre souhait de nous inspirer des systèmes vivants pour la réalisation de notre architecture. Par cette inspiration nous espérons conférer à notre architecture les propriétés d'adaptation et d'évolution qui ont permis aux espèces vivantes, et à l'humain en particulier, de coloniser la planète et de mener une existence pérenne.

Cependant, il ne faut pas perdre de vue que la plate-forme sur laquelle nous allons déployer notre architecture de contrôle sera radicalement différente des systèmes nerveux biologiques. Dans la sous-section 2.1.4 nous avons d'ailleurs expliqué les problèmes que l'on aurait rencontré à vouloir simuler un système massivement parallèle, comme le système nerveux humain, avec un calculateur artificiel aux capacités de calcul limitées. En effet, le fait que tous les processus impliqués dans un raisonnement aient à s'exécuter sur un nombre fini de processeurs présente un gros goulot d'étranglement. Dans cette section, nous allons donc présenter des architectures qui ont fait leurs preuves en informatique et en robotique afin de choisir au mieux quels compromis faire entre recopier les systèmes biologiques et utiliser d'imposants modèles mathématiques et logiques.

3.1 Des architectures de contrôle

Considérons un robot qui disposerait de toutes les connaissances imaginables : s'il n'est pas capable de les utiliser au bon moment, elles ne lui servent à rien. Pire : s'il résout un problème de peu d'importance alors qu'une tâche prioritaire doit être exécutée, cela peut l'amener dans un état critique. Par exemple, si je traverse une route et que je m'aperçois que mes lacets sont défaits, il est plus judicieux de rejoindre le trottoir d'en face plutôt que de lacer mes chaussures au milieu de la route. Il est donc important de sélectionner la bonne action à entreprendre pour un contexte (état externe - état interne) donné.

Nous allons ainsi présenter dans cette section les mécanismes de *sélection d'action* les plus répandus et référencés dans la littérature. Ces mécanismes permettent de choisir, parmi tous les comportements qui s'exécutent en parallèle, celui qui aura l'exclusivité d'accès aux actionneurs.

3.1.1 Subsomption

Pour s'assurer de l'exécution des comportements les plus importants par rapports aux autres comportements, Rodney Brooks propose en 1986 l'utilisation de mécanismes de subsomption [Brooks, 1986]. Ce type d'architecture reste aujourd'hui encore une référence dans le domaine des systèmes multi-agents. La caractéristique principale de ces mécanismes est de hiérarchiser les comportements afin de permettre à un comportement de haut niveau d'agir sur des comportements de plus bas niveau s'exécutant en parallèle.

Réaliser une architecture de subsomption consiste donc à paralléliser les tâches. On effectue alors un découpage en couches comportementales comme sur la figure 3.1. Chaque couche est reliée au capteurs et aux actionneurs et exprime un comportement particulier, comme l'évitement d'obstacle, ou une compétence spécifique, comme la locomotion. On peut alors exprimer la réalisation d'une tâche complexe en décomposant celle-ci en plusieurs comportements réactifs, ce qui facilite la programmation des comportements. Un autre bénéfice de cette architecture provient du découpage en couches comportementales, et en modules autonomes. Les modules étant autonomes, une panne sur l'un des modules ne se généralisera pas à l'ensemble du système. En effet, après la perte d'un module, le système continuera de fonctionner, même en mode dégradé.

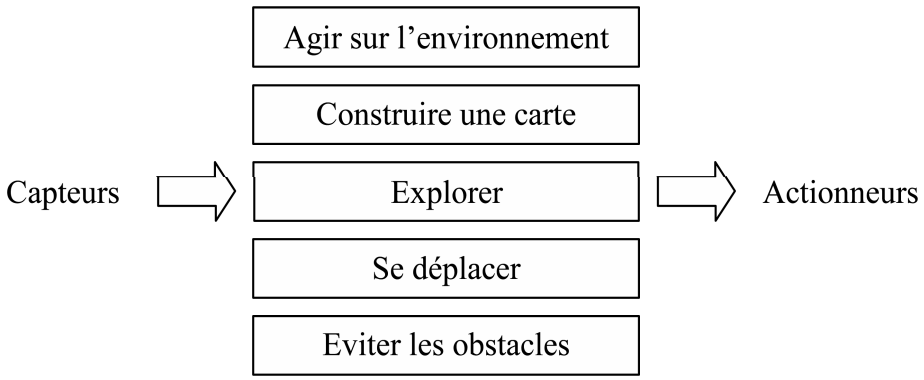

FIG. 3.1 – Décomposition de la structure de contrôle d'un robot mobile basée sur le comportement d'accomplissement de tâches.

Pour gérer cette forme de hiérarchie entre les comportements, Brooks prévoit un mécanisme de communication liant les comportements de rang supérieur à ceux de rang inférieur. Cette communication permet aux comportements de rang supérieurs d'agir sur les comportements de rang inférieur selon un principe de **forçage et inhibition**

illustré par la figure 3.2. Les entrées d'un comportement peuvent être supprimées ou modifiées pendant une certaine durée. On peut également choisir d'inhiber les sorties pendant un certain laps de temps. La structure finale comme celle présentée sur la figure 3.3 aura alors la possibilité de forcer l'exécution d'une tâche vitale ou bien d'inhiber les commandes de comportements non prioritaires. De cette manière, l'expression du comportement d'exploration pourra être inhibée au profit du comportement de retour à la base lorsque le niveau de charge des batteries devient critique.

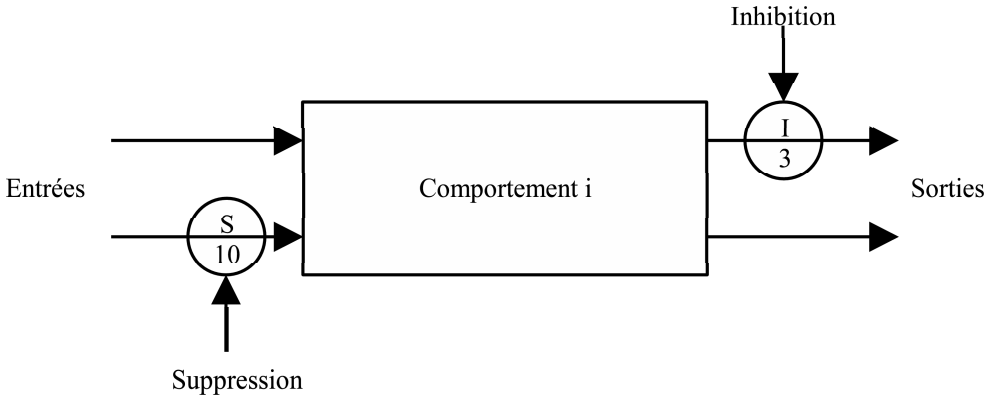

FIG. 3.2 – Processus d'interaction entre comportements. Les signaux en entrée peuvent être supprimés, les signaux en sortie peuvent quant à eux être inhibés.

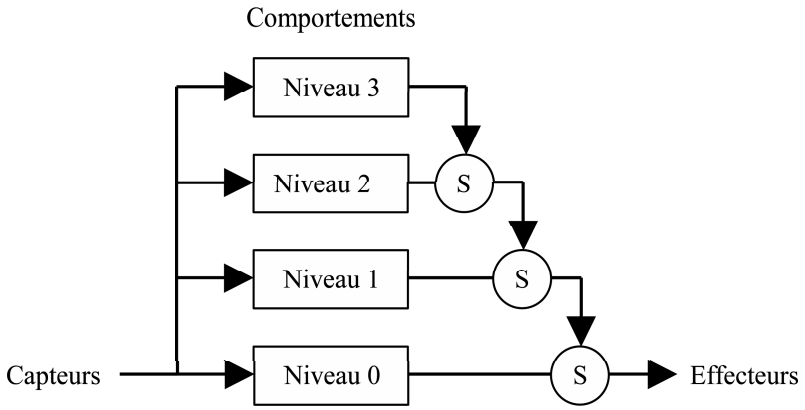

FIG. 3.3 – Hiérarchie entre comportements au sein d'une architecture à base de subsomption.

De nombreuses architectures de contrôle définissent une hiérarchie entre comportement en se basant sur le principe de subsomption. Parmi les travaux les plus répandus dans la littérature, on peut citer ceux de Matarić [Matarić, 1992], ou encore se pencher sur le mécanisme de sélection d'action de l'architecture satisfaction / altruisme de Simonin [Simonin, 2001] que nous verrons plus tard. Néanmoins les architectures

de contrôle à base uniquement de subsomption présentent une certaine rigidité problématique liée au fait que la hiérarchie entre comportements est fixée une fois pour toutes. D'autres travaux, comme ceux de [Connell, 1990], proposent d'utiliser un ordre hiérarchique partiel plutôt qu'un ordre total comme proposé par Brooks. Ainsi plusieurs comportements peuvent être au même niveau hiérarchique et peuvent donc être considérés comme concurrents, ce qui oblige à prévoir des mécanismes permettant de résoudre les conflits entre les comportements. On peut également citer le principe de **sélection dynamique** de Pattie Maes [Maes, 1991] qui contrairement à Brooks ne définit aucune hiérarchie entre comportements mais laisse les liens activateurs / inhibiteur existants produire une sorte de *sélection dynamique*. Cependant ce type de mécanisme peut produire des oscillations entre deux comportements concurrents voire poser des problèmes d'interblocage (deadlocks) lorsque deux comportements s'annulent mutuellement.

C'est pourquoi nous allons présenter directement l'**EthoModélisation** proposée par Drogoul qui prévoit, entre autres, l'existence d'une tâche par défaut activée si aucun comportement n'est actif.

3.1.2 EthoModélisation

L'éthologie est la science qui étudie le comportement des êtres vivants : animaux et êtres humains. Pour l'éthologue, tous les comportements ont une base physiologique : un comportement est produit par les différents stimuli qui affectent le sujet. Les stimulations peuvent être *endogènes*, quand elles sont internes à l'individu, ou *exogènes* quand elles sont externes à l'individu et proviennent de l'environnement. De plus, parmi ces comportements on oppose les comportements innés qui dépendent du patrimoine génétique de l'espèce et qu'on appelle *l'instinct*, aux comportements acquis qui sont les résultats d'expériences, d'apprentissage et de raisonnements et que l'on appelle *réflexes conditionnés*. L'étude du comportement des animaux apporte plusieurs bénéfices :

- Cela nous permet d'optimiser la protection d'espèces menacées en déterminant les facteurs qui entraîneraient l'extinction d'une espèce puis en régulant les quotas de chasse et de pêche.
- Les exploitants de bétail peuvent aussi améliorer les conditions de vie et les performances des animaux qu'ils élèvent augmentant ainsi les taux de rendement.
- Enfin, l'étude des comportements de certaines espèces nuisibles permet de les combattre plus efficacement.

Les bénéfices apportés par l'éthologie ne se limitent pas à l'étude des espèces vivantes. En effet, lorsque l'on souhaite réaliser une entité artificielle « vivante », on attend d'elle qu'elle ait certains comportements et on peut choisir de réaliser la structure de contrôle de cette entité comme un ensemble de comportements évoluant en parallèle et mis en concurrence pour l'accès aux effecteurs. Ainsi, en informatique, Drogoul s'est

intéressé à la manière dont les éthologues formalisent le déclenchement et l'activation des comportements. En 1993, il s'inspire de cette science et propose l'*EthoModélisation* [Drogoul, 1993] comme méthode permettant la modélisation d'un système de contrôle. Il a également implémenté un cadre d'applications destiné au développement d'architectures de contrôle basées sur l'EthoModélisation : l'EthoModelling Framework (EMF).

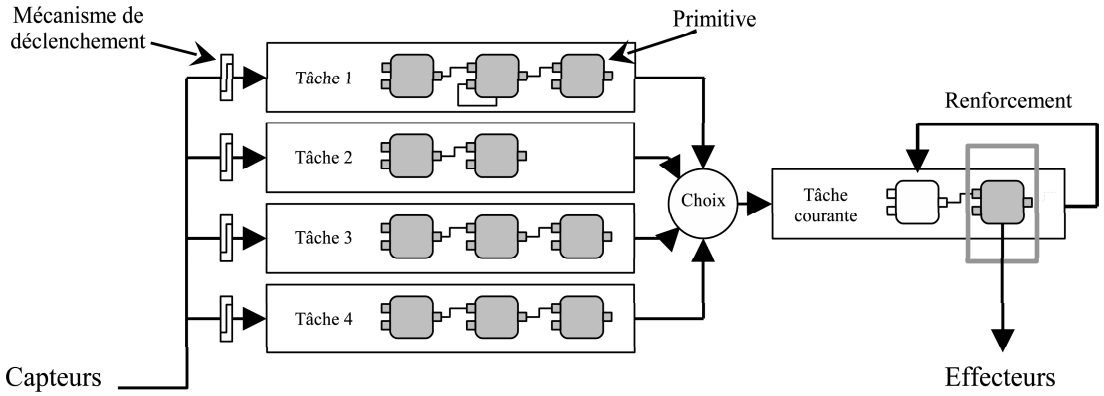

FIG. 3.4 – Illustration d'une architecture *EMF*.

L'architecture de contrôle proposée (cf. figure 3.4) est constituée d'un ensemble de tâches indépendantes qui caractérisent le comportement de l'animal. Chacune de ces tâches est une composition de *primitives*, c'est à dire de comportements moteurs élémentaires. Le déclenchement d'une tâche dépend d'une part d'une stimulation (endogène ou exogène) à laquelle est associée une valeur représentant la force de cette stimulation, et d'autre part d'un degré de motivation qui représente la force de déclenchement de la tâche. Ce degré de motivation est calculé à partir de plusieurs facteurs propres à la tâche : un *seuil inhibiteur*, un *poids* et un *niveau d'activité*. *Le poids* représente l'importance de la tâche pour le système : si le poids d'une tâche donnée est élevé cela signifie que le système se spécialise dans l'exécution de cette tâche. Cette valeur pourra être modifiée par le processus de renforcement. Le *seuil inhibiteur*, ou seuil de déclenchement, est une valeur en-dessous de laquelle la tâche ne sera pas déclenchée par un stimulus. Lors du processus de sélection d'action, les seuils de déclenchement des tâches non sélectionnées diminuent alors que le seuil de la tâche active augmente.

Enfin, le niveau d'activité est calculé lorsque la tâche est active. Il est initialisé avec le niveau d'activation de la tâche (qui est le produit de la force du stimulus déclencheur et du poids de la tâche) puis est décrémenté tant que la tâche est active. Lorsque ce niveau d'activité arrive à zéro, la tâche est stoppée.

Le processus de sélection d'action consiste alors à :

– Collecter les stimuli et éliminer ceux qui ne correspondent à aucune tâche.

- Sélectionner les tâches activables. Une tâche devient activable lorsque le niveau d'activation de cette tâche est supérieur à son seuil de déclenchement et également supérieur au niveau d'activité de la tâche en cours d'exécution.
- Activer la tâche, parmi les tâches activables, celle dont le niveau d'activation est le plus élevé après avoir pris soin de désactiver la tâche courante.

Les bénéfices apportés par cette méthode de modélisation sont nombreux, et nous nous sommes particulièrement intéressés à deux aspects importants de cette modélisation.

En premier point, ce modèle permet l'écriture du comportement d'un agent comme un ensemble de tâches indépendantes composées de séquences de comportements élémentaires appelées primitives. Cette approche facilite l'écriture de la structure de contrôle des robots : en effet pour une famille de robots dont la structure physique est proche les comportements moteurs élémentaires n'auront à être écrits qu'une seule fois. On pourra ensuite composer ces primitives comportementales pour obtenir différentes structures de contrôles permettant d'adapter un robot aux tâches qu'il aura besoin de remplir.

Le deuxième aspect qui nous intéresse réside dans le processus de sélection d'action, et plus particulièrement aux valeurs et poids utilisés. En effet, les stimuli à force variable combinés aux motivations internes de l'individu permettent un déclenchement de tâches qui ne dépend pas uniquement des perceptions à un moment donné mais également de l'expérience antérieure de l'agent dans cette tâche. La variation du poids associé à une tâche permet ainsi de représenter des phénomènes de renforcement ou d'habituation comme ceux observés chez les animaux. Le processus de sélection d'action assure ainsi un basculement efficace entre les tâches connues du robot lui permettant d'exprimer ses comportements en réponse aux stimulations de l'environnement tout en prenant en compte l'expérience accumulée pour chaque tâche.

Cependant nous aimerions étendre la portée de cette modélisation à des systèmes apprenants. En effet, le principal apport de l'éthomodélisation est de permettre à un humain de réaliser la structure de contrôle d'un robot en exprimant les différents comportements qu'il souhaite obtenir. L'étape suivante serait donc de permettre au robot de construire lui même les séquences de primitives lui permettant de réaliser les tâches dont il a besoin[1]. Enfin si une tâche est une composition de primitives, pourquoi ne pas imaginer différents niveaux de composition permettant de considérer une tâche de niveau n comme une primitive pour les tâches de niveau $n+1$. Dans un processus d'apprentissage, cela permettrait au robot de se construire des comportements de plus en plus complexes en s'appuyant sur les connaissances qu'il aura déjà acquises depuis sa *naissance.*

[1] Il faudra également définir cette notion de besoin

3.1.3 Discussion

Problèmes communs à ces modèles :
- de l'adaptation est réalisée, mais pas forcément de l'apprentissage
- comment s'affranchir d'un superviseur humain ?

3.2 Des architectures émotionnelles

Les émotions sont souvent considérées comme ce qui différencie les êtres vivants des êtres artificiels. Lorsque je présente mes travaux de recherche à des amis non informaticiens, beaucoup sont perplexes sur le fait qu'une entité artificielle puisse éprouver des émotions. Il est vrai que les robots que l'on utilise n'ont pas besoin de ressentir ou d'exprimer des émotions pour réaliser la tâche pour laquelle ils ont été conçus. Mais quelle en est la raison ? Pour répondre à cette question, intéressons-nous à ce que les émotions apportent aux êtres vivants, et en particulier aux humains et aux primates. Pour William James, l'émotion est le sentiment qui résulte des changements d'état corporels produits par une perception [James, 1884]. Selon Darwin, l'expression faciale des émotions primaires comme celles montrées sur la figure 3.5 telles que la colère, la peur, la joie, est innée et universelle. Pour argumenter cette proposition, il suffit d'observer l'expression faciale d'un bébé lorsque l'on porte à ses lèvres une substance sucrée : le bébé exprimera la joie avec une grande spontanéité.

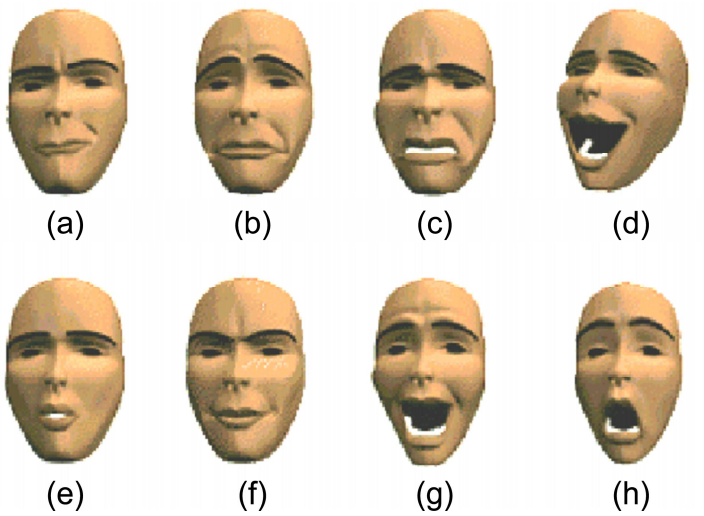

Fig. 3.5 – Expressions faciales de diverses émotions de base. (a) doute, (b) tristesse, (c) dégoût, (d) joie, (e) inquiétude, (f) courroux, (g) étonnement, (h) horreur.

Mais dans quels buts ces émotions seraient universelles et innées ? Les sociologues nous apportent une réponse en s'accordant à définir les émotions comme des formes

élémentaires d'expression nécessaires à une vie sociale. En effet, pour que la cohabitation de plusieurs individus soit possible au sein d'une société, il faut que des moyens de communications élémentaires et universels permettent de résoudre les conflits que l'on rencontre en société. L'expression de la colère sur le membre d'un groupe signifie aux autres qu'il y a un conflit potentiel à régler, et l'expression faciale correspondante permet de transmettre le message rapidement et sans ambiguïté. De la même manière, lorsqu'un membre du groupe est satisfait par une certaine situation, en exprimant la joie il signifie aux autres que l'interaction qu'il a avec eux est satisfaisante. Il renforce ainsi dans l'esprit de ses congénères le fait que les actions qu'ils ont produites sont bénéfiques pour lui, et les encourage à les renouveler. Le fait que ces expressions faciales soient innées et universelles permet de se passer de moyens de communication plus évolués comme les actes de langage pour améliorer les interactions entre plusieurs individus de culture et de capacités cognitives différentes. Cela permet lorsque l'on soigne des nouveaux nés d'estimer leur souffrance en se basant sur l'expression de leurs émotions pour leur donner la dose nécessaire et suffisante de médicaments atténuant la douleur.

3.3 Architecture centralisée ou distribuée ?

Il est nécessaire pour élaborer la structure de contrôle d'un robot de déterminer si l'intelligence sera centralisée et détenue par une unique entité, ou si cette intelligence sera distribuée sur une multitude d'entités en interaction. Nous présenterons donc dans cette section les spécificités de ces deux types d'architecture.

3.3.1 Architectures centralisées

On dit qu'une architecture de contrôle est centralisée quand toutes les décisions sont prises par une seule entité. Pour déterminer le comportement de l'intégralité de l'architecture et les actions à entreprendre, cette entité centrale va alors superviser et contrôler l'exécution des autres entités qui composent l'architecture. Les verbes « contrôler » et « superviser » sont à prendre ici au sens de « maîtrise » et non de « vérification », comme le souligne Jacques Ferber en introduction de la version française de son livre [Ferber, 1995]. En effet pour souligner les relations au sein d'une architecture centralisée, on emploiera les termes de maître / esclave ou encore superviseur / supervisé pour décrire les entités qui la composent.

En informatique, on peut prendre comme exemple d'architecture centralisée les systèmes experts. Ce sont des applications informatiques destinées à remplacer l'humain dans des tâches qui nécessitent des raisonnements complexes et beaucoup d'expériences dans le domaine concerné par le système expert. Dans un tel système, une entité unique fait appel aux différents objets et méthodes dont il dispose et pilote les différentes acti-

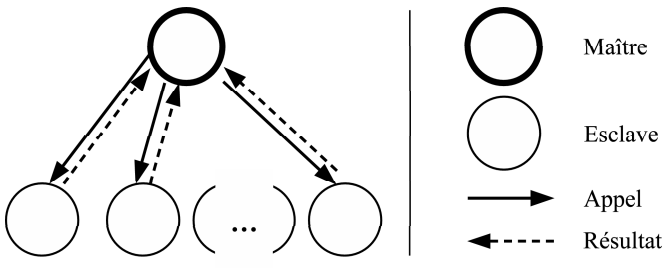

FIG. 3.6 – Représentation d'une architecture centralisée. Dans ce type d'architecture les esclaves ne prennent pas d'initiative, à l'opposé du maître qui les « interroge » successivement pour prendre ses décisions.

vités du système pour produire le fruit de son raisonnement. Il existe aussi en robotique mobile distribuée des exemples d'architectures de contrôle centralisées.

La figure 3.7 montre l'utilisation d'une telle architecture pour la catégorie des micro-robots footballeurs. L'équipe est constituée de robots mobiles qui disposent des capacités d'action. Pour des raisons d'encombrement, ils ne disposent que de capacités perceptives et cognitives limitées. L'équipe est pilotée par un ordinateur qui est équipé d'une ou plusieurs caméras ainsi qu'une puissance de calcul plus importante que celle embarquée sur les robots. La pièce centrale de cette équipe, à savoir l'ordinateur, utilise les moyens de perception du système pour l'élaboration de la tactique et le choix des actions. Les robots ne représentent alors que les capacités d'action du système et sont esclaves de l'ordinateur.

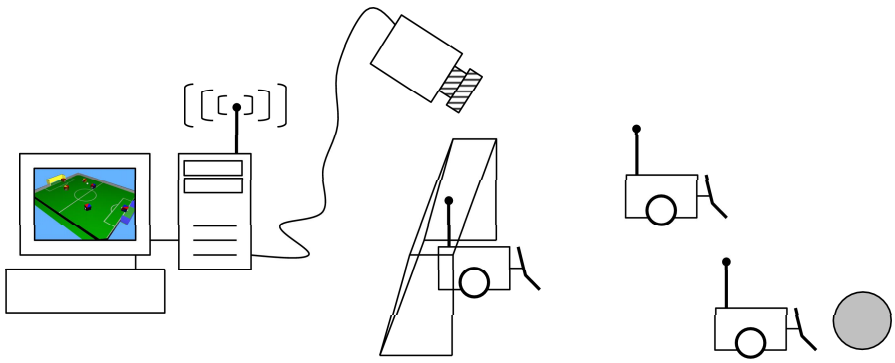

FIG. 3.7 – Utilisation d'une architecture centralisée dans le cadre d'une application de robots footballeurs.

Ce type d'architecture centralisée présente plusieurs avantages. En outre cela permet de concentrer l'intelligence sur l'entité centrale permettant ainsi de piloter des entités plus rudimentaires ne disposant pas de puissance de calcul. Un autre avantage est que la programmation de l'intelligence de ce système est plus facile pour un humain car

celui-ci ayant déjà le rôle de programmateur et d'administrateur s'imagine facilement dans le rôle du superviseur.

Mais ce type d'architecture amène aussi des problèmes non négligeables, en particulier en ce qui concerne la robustesse du système. Dans la nature, si l'on tue la reine d'une colonie d'abeilles, la ruche est amenée à disparaître. Dans l'exemple des robots footballeurs, si l'on supprime le superviseur, le système se fige, les robots n'étant dotés d'aucune autonomie décisionnelle. La robustesse du système est également compromise en cas de problèmes de communication entre le maître et les esclaves. Si on considère des véhicules d'exploration de la planète Mars, il ne serait pas judicieux de centraliser l'intelligence sur terre, les communications mettant plusieurs minutes pour atteindre la planète rouge. Il apparaît donc nécessaire de déporter, en tout ou partie, l'intelligence sur les différents acteurs du système.

3.3.2 Architectures décentralisées-distribuées

Face aux problèmes de robustesse engendrés par la centralisation du contrôle, la perspective de déporter l'intelligence sur tous les acteurs du système apparaît comme une solution intéressante. Ainsi, dans une architecture décentralisée, le contrôle est distribué sur des entités qui composent le système. Ces entités doivent alors se coordonner et coopérer à la réalisation de la tâche demandée au système. La distribution du contrôle permet alors de casser la complexité inhérente au système. En effet, il est plus difficile d'obtenir un modèle exact et simple d'un système complexe plutôt que de décrire la structure et le comportement des entités qui le composent ainsi que les interactions entre ces entités.

Pour réaliser une telle architecture, les entités qui composent le système doivent avoir une autonomie décisionnelle ainsi que des capacités de communication. En effet, dans une architecture distribuée aucune entité n'a de vue globale du système et cela soulève plusieurs problèmes :

- Une entité n'a pas de contrôle direct sur les autres entités et il n'y a pas d'entité qui supervise les autres. On doit donc mettre en place des méthodes de coopération et de coordination afin de d'éviter les conflits et les situations de blocage.
- Une entité ne connaît pas l'état global du système, les décisions sont donc prises de manière locale. Il en résulte des décisions qui sont optimales localement, ce qui amène le système à produire des solutions qui ne sont pas forcément optimales globalement.

La figure 3.8 montre l'exemple d'un groupe de robots qui permet de résoudre de manière distribuée un problème de fourragement. Le but du système est de transporter l'intégralité de la matière en vert vers la zone de dépôt représentée par l'arc de cercle rouge en haut à droite. L'exercice se termine lorsque tout l'environnement a été nettoyé.

Le système est hétérogène : certains robots peuvent effectuer des découpes, les autres n'ont que la capacité à déplacer les objets. Les deux types de robots ne communiquent que par quelques signaux leur permettant de résoudre des conflits et améliorer leurs comportements comme décrit dans [Chapelle *et al.* , 2002].

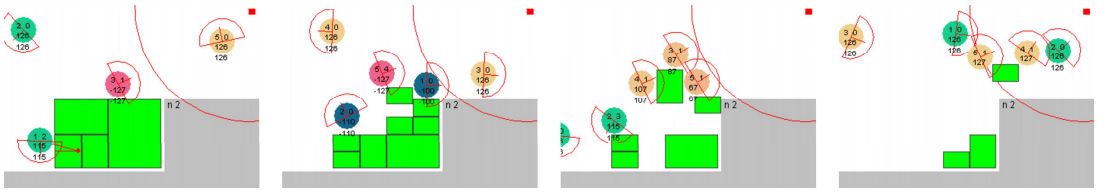

FIG. 3.8 – Utilisation d'une architecture distribuée dans le cadre d'une application de robots fourrageurs hétérogènes et apprenants.

Dans une architecture distribuée, la solution au problème est trouvée lorsque l'environnement se stabilise. Cette solution résulte des interactions entre les agents qui composent le système. On parle alors de l'émergence d'une solution provenant d'une *intelligence collective* [Ferber, 1995].

3.3.3 Discussion

On ne peut se prononcer sur la supériorité d'un type d'architecture sur l'autre. En effet, chaque type d'architecture, centralisée ou distribuée, présente des avantages et des inconvénients.

Lors de l'analyse des besoins, adopter une approche descendante[2] revient à s'intéresser aux résultats et au comportement du système pour en écrire un modèle qui produise les résultats voulus. Le déploiement de ce type d'architecture mènera souvent à une architecture centralisée. Ce genre d'architecture nous semble plus facile à imaginer, et cela peut s'expliquer par notre manière de considérer notre environnement comme une structure très hiérarchisée dans laquelle on retrouve souvent le modèle superviseur / supervisé. Ce modèle se retrouve appliqué à d'autres domaines comme les organisations sociales et administratives, ainsi qu'à des processus de production dans l'industrie.

Cette vision centralisée implique que le superviseur de l'architecture ait une connaissance la plus globale possible du reste du système afin de pouvoir déterminer les actions que les supervisés vont devoir exécuter. La conséquence directe est que dans le cadre d'une architecture complexe faisant intervenir de nombreux éléments, le superviseur devra posséder de grandes capacités de stockage pour mémoriser l'état de l'environnement et de ses supervisés. Il devra également disposer d'une grande puissance de calcul pour animer tous ces sujets, ainsi que des algorithmes lui permettant de résoudre les

[2]Top-Down en anglais

problèmes de conflit et de blocage. On devra également s'assurer que le superviseur ne tombera pas en panne, car tout le système se bloquerait.

En terme d'analyse, on peut faire le parallèle avec une petite ou moyenne entreprise dans laquelle le chef d'entreprise a une vue globale de son entreprise et de son environnement, demande à ses employés de réaliser des parties du travail et règle les conflits entre les employés. Cependant, pour ce qui est du déploiement de cette architecture, la comparaison s'arrête là, car même s'il existe une hiérarchie au sein de l'entreprise, chacun des employés est en réalité autonome et peut décider de ne pas faire ce qu'on lui a demandé, voire de prendre des initiatives dans son travail. Cela est encore plus vrai dans une grande entreprise où le Président Directeur Général ne peut avoir une vue complète de son entreprise et du système de production. De plus, en cas de disparition du PDG, le système ne se bloque pas et une nouvelle personne prend la direction de l'entreprise. Néanmoins si l'on considère, comme à l'armée, que chaque entité du système ne prend pas d'initiative et exécute les ordres qui lui sont donnés, cette architecture est dite centralisée.

Adopter une approche ascendante[3] consiste à s'intéresser aux différentes entités qui constituent un système complexe. Plutôt que de chercher à décrire le comportement général du système, on essaye de modéliser les différents acteurs du système et leurs interactions entre eux. Ce genre d'approche conduit donc à une architecture distribuée dont le comportement global résulte des interactions entre les entités qui la composent. Le fait de distribuer l'intelligence sur plusieurs entités plus simples et éventuellement redondantes est un gage de robustesse pour le système. En effet si une des entités tombe en panne, le système peut continuer à fonctionner, même en mode dégradé.

Pour illustrer les architectures centralisées dans le paragraphe précédent nous avons utilisé l'image d'une structure militaire : l'armée. Si l'on veut poursuivre sur la même catégorie de métaphores, on peut illustrer les architectures distribuées en faisant le parallèle avec les guérillas. Une guérilla décrit une organisation menant des combats en petits groupes, mobiles et flexibles. Il devient alors beaucoup plus difficile de combattre ou d'empêcher une telle structure d'agir, car même en supprimant les entités les plus influantes du réseau, la structure peut se réorganiser et continuer à fonctionner. La guerre menée par les Etats-Unis contre l'organisation Al Quaïda est un exemple contemporain qui met en évidence la supériorité, en terme de robustesse, d'une architecture distribuée. En effet face à une armée ayant de lourds moyens techniques et logistiques, mais disposant d'un commandement centralisé et hiérarchisé, une structure distribuée constituée d'entités disposant de faibles ressources se montre plus résistante.

Revenons au sujet qui nous intéresse. Nous avons exprimé le besoin d'obtenir des structures de contrôle qui confèrent à un robot les facultés suivantes :

[3]Bottom-Up en anglais

- rester en vie : en assurant son autonomie, et en donnant satisfaction à son concepteur pour éviter d'être rangé au placard
- s'adapter à sa structure physique ainsi qu'à son l'environnement
- évoluer pour prendre en compte des modifications de sa structure (ajout ou suppression de composants matériels) et pour apprendre des tâches de plus en plus complexes.

Pour cela, nous avons choisi de nous inspirer d'un système complexe : le cerveau humain. Plus particulièrement, nous avons eu une approche ascendante en nous intéressant à la structuration du cortex en unités élémentaires de calcul. Comme nous l'avons vu dans cette section, une vision distribuée permet de reproduire avec plus d'aisance des systèmes complexes. Nous avons également souligné l'intérêt des architectures distribuées pour la réalisation de structures de contrôle robustes aux perturbations et aux évolutions. C'est donc tout naturellement que nous avons choisis de concevoir la structure de contrôle du robot comme une architecture distribuée.

Chapitre 4

Un modèle d'architecture de contrôle

DANS les chapitres précédents, nous avons exposé les objectifs que nous souhaitons faire atteindre à notre modèle d'architecture de contrôle. Pour cela nous avons présenté dans le chapitre 2 les structures de contrôle ayant fait leurs preuves dans le monde du vivant, et nous nous sommes plus particulièrement intéressés à la structure du système nerveux humain et à son fonctionnement. Puis, dans le chapitre 3 nous avons présenté différentes architectures de contrôle utilisées en Intelligence Artificielle et en robotique collective. Enfin, nous avons terminé en montrant notre intérêt pour les architectures de contrôle à base d'agents. Nous avons donc déjà vu divers systèmes multi-agents, notamment dans la sous-section 3.3.2 qui présentait une architecture distribuée où chaque robot était un agent. Dans ce chapitre, nous allons présenter l'architecture de contrôle d'un robot que nous avons conçu. A partir de maintenant, nous nous focaliserons sur le comportement d'un seul robot et non d'un groupe de robots. Ainsi le terme « système multi-agents » ne désigne plus l'ensemble des robots, mais la structure de contrôle d'un robot : « son cerveau ». Le terme « agent » désigne donc une des entités composant la structure de contrôle du robot.

Dans ce chapitre, nous commencerons par définir, en section 4.1, les outils que nous utiliserons pour réaliser et animer l'architecture de contrôle. Puis, dans un second temps, nous proposerons en section 4.2 le système multi-agents permettant de réaliser l'architecture de contrôle. Une fois que nous aurons vu comment s'articule cette structure de contrôle nous pourrons passer au chapitre 5 qui présentera le processus d'apprentissage permettant l'émergence de la structure de contrôle adéquate et son adaptation aux besoins du système.

4.1 Les outils de modélisation

4.1.1 Les émotions : motivations de l'apprentissage

> "*La nature a donné aux organismes animés, doués de mouvement, la faculté de sentir la douleur - afin de préserver les membres susceptibles d'être amoindris ou détruits dans l'accomplissement de ces mouvements*"

(Manuscrit de la Bibliothèque de l'Institut de France - H60 (12)r. Léonard de Vinci)

Quand on réalise un système apprenant, on cherche à obtenir un système qui choisisse par lui-même la politique qui lui permettra de maximiser les récompenses qu'il obtient. Dans le cas d'un apprentissage supervisé, le professeur observe l'élève, et estime l'efficacité des actions qu'il fait. Dans cette configuration le but de l'apprenant est de produire les comportements qui satisferont le professeur. De nombreux travaux sur les émotions ont été conduits dans divers domaines comme en neurosciences [Damasio, 1994], en informatique [Minsky, 1986] [Shibata *et al.* , 1996] ou en psychologie. Ces recherches ont également donné lieu à des expérimentations comme pour le modèle DARE [Maçãs *et al.* , 2001], et à des applications commerciales comme les chiens Aibo de Sony présentés sur les figures 4.1 et 4.2 ci-dessous.

FIG. 4.1 – Le chien Aibo en pleine activité de jeux.

FIG. 4.2 – De gauche à droite : L'ERS 110, première version commerciale produite en nombre limité en 1999. L'ERS 210, produit en série en 2000. L'ERS 220 à l'allure futuriste, 2001.

Dans un système apprenant autonome, ou non supervisé, il faut donc trouver les motivations qui vont permettre de structurer l'apprentissage. Ceci afin de permettre au système de focaliser son attention, et donc son apprentissage, sur une partie du problème. Dans notre travail, nous cherchons les motivations dont l'entité apprenante a besoin pour apprendre par elle-même et développer ses connaissances, afin que l'intelligence émerge de motivations relativement simples, mais qui ont une signification forte pour l'entité apprenante. Pour arriver à cela, il est nécessaire que l'entité vive suffisamment longtemps, et rencontre beaucoup de situations différentes pour qu'elle développe ses connaissances. En conséquence, nous devons fournir à l'entité apprenante les motivations qui vont lui permettre de continuer à évoluer, tout en respectant les contraintes de l'environnement et de son propre corps. Pour réaliser un robot social et joueur, le chien Aibo [Sony, 2006] a été doté d'émotions comme l'affection, l'exercice, l'appétit et le sommeil. Par exemple, l'exercice peut être calculé par une estimation de l'énergie consommée pour réaliser des mouvements ; pour la faim, la valeur de l'émotion peut être calculée en considérant le niveau de charge des batteries. Le comportement attendu est que le robot joue jusqu'à ce que ses besoins énergétiques deviennent critiques, le forçant à chercher la base pour recharger ses batteries, puis retourner jouer à nouveau. C'est ce genre de motivations que nous souhaitons utiliser pour nos robots afin de les guider dans leur processus de sélection d'action ainsi que dans le processus d'apprentissage.

En ce qui nous concerne, pour calculer les émotions nous nous inscrivons dans le cadre du modèle Satisfaction-Altruisme [Simonin, 2001] dont les travaux s'inspirent et s'appuient sur l'observation de comportements animaux. Comme pour le modèle Satisfaction-Altruisme, nous distinguons deux types de motivations relatives à :

- l'*état interne*, qui est une évaluation des valeurs critiques de certaines perceptions internes : instinct de survie (santé, intégrité physique), faim (nécessité de recharger les accumulateurs par exemple), besoin de jouer, curiosité. Par exemple, la curiosité semble être un atout pour assurer un bon développement. En apprentissage par renforcement, cela se traduit par l'utilisation d'un coefficient d'exploration qui incite l'entité apprenante à explorer les situations qu'elle ne connaît pas.
- l'*état externe*, qui est une composition des niveaux de satisfaction donnés par les robots se trouvant dans le voisinage de l'entité apprenante. Ceci afin de réaliser un apprentissage social. Dans ce cas l'approche peut s'inspirer des comportements animaux. En effet la plupart des espèces animales comprennent les mêmes expressions faciales qui permettent de communiquer des émotions simples comme la joie, la peur, la colère. D'autres travaux qui parlent d'apprentissage social ont montré qu'une architecture où les robots apprennent à satisfaire leurs voisins permet d'obtenir une meilleure coopération [Chapelle *et al.* , 2002] [Matarić, 1994b].

Dans le modèle Satisfaction-Altruisme, la satisfaction personnelle (figure 4.3) est définie comme une valeur qui représente, à un instant donné, l'état de progression de la tâche courante. Dans cette approche, à chaque tâche est associée une politique régis-

sant l'évolution de la valeur de satisfaction personnelle. Cela consiste à augmenter ou à diminuer la valeur de satisfaction en fonction d'une mesure régulière de l'avancement de la sous-tâche courante. La satisfaction personnelle s'exprime alors dans un intervalle borné $[-P_{max}, P_{max}]$. Sur cet intervalle on peut définir des seuils permettant d'abandonner une tâche ou d'émettre un signal. Dans les travaux qui ont conduit à cette thèse [Chapelle, 2001], nous avions justement définis deux seuils ($sEm+$ et $sEm-$) au delà desquels le robot procède à l'émission d'un signal d'attraction ou de répulsion.

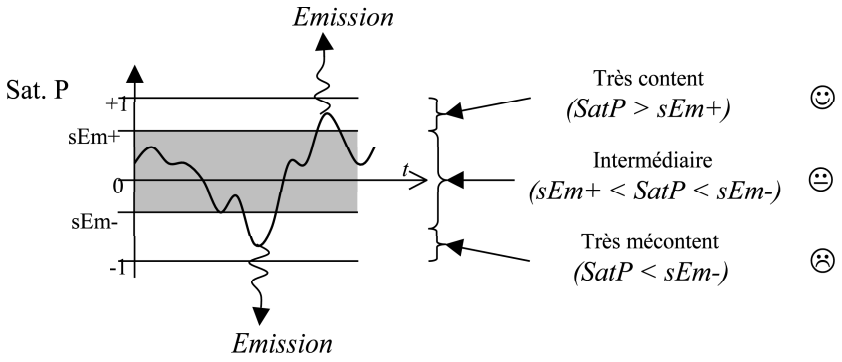

FIG. 4.3 – Exemple de courbe de satisfaction personnelle.

Le signal émis, appelé *signal de satisfaction interactive*, représente l'état externe du robot. Il a pour but d'induire une réaction chez les autres robots suffisamment proches pour pouvoir percevoir le signal émis. Cette réaction peut se traduire chez les autres robots par leur basculement dans un état altruiste, état dans lequel ces robots vont entreprendre des actions dans le but de résoudre un conflit ou d'aider le robot à l'origine de l'émission du signal. Chaque robot dispose d'une valeur représentant une mesure de la qualité d'interaction appelée *satisfaction interactive* et qui est en fait la valeur du signal le plus fort perçu (on notera qu'un signal négatif est prioritaire sur tous les signaux positifs). Le *coefficient d'altruisme* d'un robot définit alors la proportion entre la satisfaction personnelle et la satisfaction interactive au delà de laquelle le robot basculera en état altruiste.

Les travaux d'apprentissage réalisés dans le cadre du modèle Satisfaction-Altruisme [Chapelle, 2001] [Chapelle *et al.* , 2002] nous ont montré les bénéfices apporté par l'utilisation des concepts de satisfaction personnelle et de satisfaction interactive dans le processus de sélection d'action et de renforcement des poids de déclenchement associés aux actions coopératives. Suite à ces travaux, nous avons voulu proposer une méthode d'apprentissage plus « souple ». En effet nos précédents travaux permettaient d'optimiser un processus de sélection où les actions étaient écrites par le concepteur. Or nous souhaitions réaliser une architecture de contrôle qui soit capable par apprentissage de construire elle même les composantes de sa structure de contrôle. Mais le fait que

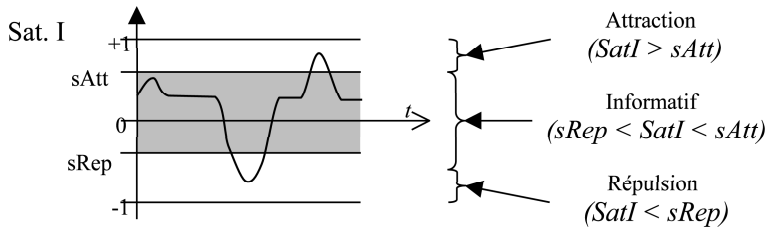

FIG. 4.4 – Exemple de courbe de satisfaction interactive. Dans cet exemple deux seuils *sAtt* et *sRep* définissent une force de signal au delà de laquelle sera déclenché une attraction ou une répulsion vers le robot émettant le signal.

toutes les tâches influent sur la même motivation nous a semblé être un facteur limitant la construction de structures de contrôle. En effet, si une action peut représenter une solution à la satisfaction d'un besoin, elle peut également se traduire par une lourde insatisfaction d'un autre besoin. C'est cela qui nous a motivé à choisir de représenter **plusieurs** indices de satisfaction relatifs aux différents états internes du robot, et que nous appellerons **émotions**. Chaque *émotion* est liée à un but à atteindre et est représentée par une valeur positive dans l'intervalle $[0, 1]$. Plus cette valeur est élevée, plus l'émotion correspondante est satisfaite. A l'opposé, une valeur faible représente une situation de gêne, un état émotionnel qui nécessite de la part du robot une réaction pour satisfaire cette émotion. La valeur de l'émotion peut être obtenue de manière instantanée si le robot dispose d'un capteur adéquat. Le cas échéant, la valeur de l'émotion peut être calculée de manière incrémentale.

Définition 4.1 (Emotion) *Une émotion est, pour un robot, à un instant donné, un état fonction de l'intensité d'un besoin auquel cette émotion est liée.*

Cette définition, volontairement vague, s'appuie sur la notion de *besoin*. En effet un besoin peut être relatif à la survie du robot, comme par exemple l'état de charge des batteries, ou encore son intégrité physique. Mais un besoin peut également être relatif aux résultats que le concepteur souhaite obtenir et qui ont motivé le choix d'une architecture de contrôle émotionnelle comme celle-ci. Comme nous l'avons vu précédemment, une émotion est caractérisée par sa *valeur* : nous allons donc définir celle ci.

Définition 4.2 (Valeur d'une émotion) *Pour un besoin donné, la valeur d'une émotion i est un réel défini dans* $[0, 1]$ *qui est fonction de l'état de satisfaction de ce besoin.*

- *Plus* $emotion_i$ *est proche de* 0*, plus on dira que l'émotion i est insatisfaite.*
- *A l'inverse, quand* $emotion_i$ *est proche de* 1*, on dira que l'émotion i est satisfaite.*

Considérons l'émotion de faim dont nous avons parlé précédemment. Si la valeur de cette émotion s'approche de 1, l'état correspondant sera celui de « Satiété ». A l'opposé, quand la valeur approche de 0, l'état correspondant sera la sensation de « Faim ». Il ne faut donc pas perdre de vue que même si le nom donné à l'émotion s'apparente à une sensation négative ou à une douleur, la sémantique de la valeur 1 correspondra toujours à la notion de plaisir et la valeur 0 à une sensation de manque ou de douleur. A présent, et à partir de cette définition, nous proposons deux manières de calculer la valeur d'une émotion selon qu'il existe un capteur adéquat, ou pas. Dans le premier cas, la valeur de l'émotion est calculée instantanément à partir du capteur adéquat, alors que dans le second cas elle doit être calculée incrémentalement.

Calcul d'une émotion de manière instantanée

Lorsque le robot dispose d'un capteur spécifique permettant de mesurer une émotion, la valeur de celle-ci sera donnée par la formule 4.1. :

Soit v_i la valeur renvoyée par le capteur lié à l'émotion i,

Soient v_{min} et v_{max} les valeurs minimum et maximum que peut prendre v_i,

$$emotion_i = \frac{v_i - v_{min}}{v_{max} - v_{min}} \tag{4.1}$$

Cette équation permet de ramener l'écart de variation depuis l'intervalle $[v_{min}, v_{max}]$ vers l'intervalle $[0, 1]$. Nous aurions pu nous contenter de simplement diviser la valeur v_i par v_{max}, mais ceci n'aurait pas permis de prendre en compte des valeurs v_i négatives, et n'aurait également pas permis de considérer des cas où la valeur $v_i = 0$ n'est jamais observable.

Prenons comme exemple une émotion de faim qui serait directement liée à la charge des accumulateurs. La mesure de charge d'un accumulateur est donnée par sa capacité exprimée en mAh (milliampères par heure). La valeur v_{max} sera donnée par la capacité maximale de l'accumulateur, quant à la valeur v_{min} elle sera positionnée à la capacité minimale au delà de laquelle le robot ne peut fonctionner. En effet l'électronique du robot cessera de fonctionner avant la décharge complète des accumulateurs. Il n'est donc pas nécessaire de représenter des valeurs de charge qui ne seront jamais observées.

Calcul d'une émotion de manière incrémentale

Si l'on ne dispose pas d'un capteur permettant de mesurer la satisfaction d'une émotion donnée, on peut néanmoins synthétiser une valeur permettant de représenter l'état de satisfaction du besoin concerné. Pour cela, nous reprenons la manière dont est calculée la satisfaction personnelle du modèle satisfaction-altruisme [Simonin & Ferber, 2000]

en ramenant l'intervalle de variation à $[0,1]$. A chaque pas de temps, on effectue donc une mesure pour déterminer si l'on se rapproche ou si l'on s'éloigne de la satisfaction totale de cette émotion. Pour cela, nous allons définir plusieurs incréments positifs, ou négatifs, représentant l'évolution de l'état de satisfaction et qui seront appliqués à la valeur de l'émotion concernée. Chaque incrément est lié à une condition et sera appliqué à la valeur de l'émotion si la condition est remplie.

Soit Δt l'intervalle de temps correspondant au cycle perception - délibération - action de l'agent responsable de l'émotion i,

Soit v_i la mesure de progression vers l'état de satisfaction de l'émotion i pendant l'intervalle de temps Δt,

$$emotion_i(t) = emotion_i(t - \Delta t) + v_i \tag{4.2}$$

Avec $\forall t \geq 0$, $emotion_i(t) \in [0,1]$

Etat émotionnel d'un robot

Maintenant que les émotions sont définies, nous allons définir ce qu'est l'état émotionnel d'un robot. Pour calculer cet état émotionnel nous allons combiner les différentes valeurs de satisfaction d'un robot et en effectuer une moyenne. Cette moyenne pourra être pondérée, en effet certaines émotions peuvent avoir plus d'importance que d'autres. C'est pourquoi la pondération des valeurs des émotions peut s'avérer intéressante.

Définition 4.3 (Etat émotionnel d'un robot) *L'état émotionnel d'un robot disposant de n émotions, est défini comme la moyenne pondérée des valeurs de chacune de ses émotions :*

$$GSat = \frac{\sum_{i=0}^{n} emotion_i * poids_i}{\sum_{i=0}^{n} poids_i} \tag{4.3}$$

Si l'on considère que chaque émotion a autant d'importance que les autres, on peut choisir de ne pas pondérer les émotions. Dans ce cas, l'équation 4.3 peut se réécrire sous la forme :

$$GSat = \frac{\sum_{i=0}^{n} emotion_i}{n} \tag{4.4}$$

Emission de l'état émotionnel et interaction sociale

L'état émotionnel défini précédemment représente un synthèse des émotions du robot. Lorsque ce robot sera mis en relation avec d'autres, il sera intéressant que ces robots puissent percevoir mutuellement leurs états émotionnels. En effet, un robot doit pouvoir évaluer la pertinence de son interaction avec un autre robot et pour cela le meilleur indicateur est l'état émotionnel de ce dernier. C'est pourquoi pour permettre des comportements sociaux et un apprentissage social, l'état émotionnel d'un robot est transmis en permanence aux robots se trouvant dans son voisinage. Nous nous plaçons dans un cas particulier du modèle satisfaction-altruisme puisque le signal émis est directement *l'état émotionnel* d'un robot. Cet état émotionnel est alors utilisé par les robots qui le perçoivent comme une motivation à satisfaire, de sorte qu'ils apprennent également à satisfaire leurs voisins. Le signal correspondant à l'émission de l'état émotionnel du robot i perçu par le robot sera noté IExt_i, avec $\forall i$, $\text{IExt}_i \in [0,1]$. Plus précisément, chaque robot perçoit en permanence les états émotionnels des robots se trouvant dans son voisinage et dispose d'une émotion qui est en fait la *satisfaction interactive* du modèle satisfaction-altruisme [Simonin, 2001] ramenée à l'intervalle $[0,1]$ (équation 4.5). La valeur de cette émotion, notée EmSatI, est donnée par l'équation 4.6 dans laquelle n représente le nombre de robots dont le robot perçoit un signal.

$$\text{EmSatI} = \frac{\text{SatI} + 1}{2} \tag{4.5}$$

donc,

$$\text{EmSatI} = \text{IExt}_j \text{ , avec } \left|\text{IExt}_j - \frac{1}{2}\right| = \underset{i\,:\,1 \to n}{max} \left(\left|\text{IExt}_i - \frac{1}{2}\right|\right) \tag{4.6}$$

Il y a deux points importants dans cette équation. Tout d'abord on peut remarquer que la valeur de EmSatI est directement donnée par la valeur d'un des signaux perçus. Le signal qui est sélectionné pour cette affectation, noté IExt_j, est celui dont la valeur est le plus éloignée de la valeur $\frac{1}{2}$. En effet, de cette manière on concentre l'attention sur les signaux émis par des robots ayant un état émotionnel très insatisfaisant ou très satisfaisant.

4.1.2 L'apprentissage par renforcement

Les algorithmes d'apprentissage par renforcement [Sutton & Barto, 1998] sont très bien adaptés aux problèmes de sélection d'actions. La simplicité de ces algorithmes et leur utilisation possible dans des environnements non déterministes les ont rendus populaires et référencés dans de nombreux travaux [Matarić, 1994a] [Parker, 1998] [Shibata *et al.* , 1996].

Dans cette méthode, le robot apprend grâce à des récompenses externes reçues de l'environnement. La récompense est interprétée comme un renforcement scalaire positif ou négatif. Le but du système est d'améliorer son processus de sélection d'action afin de maximiser les récompenses externes reçues.

Le principal problème de cette approche est l'évaluation des récompenses. Dans des environnements grille simulés, les valeurs de récompense peuvent être facilement calculées [Dutech *et al.* , 2001] [Boutilier, 1999]. Dans des systèmes réels, les robots doivent percevoir par eux-mêmes la réussite ou l'échec de leurs actions. Dans notre approche émotionnelle, la réussite ou l'échec est donné par la valeur de l'émotion qui est traitée.

Pour permettre un tel apprentissage, les comportements du robot sont définis en utilisant des règles pondérées :

$$(p_i, a_i) \rightarrow (w_i, t_i)$$

Pour une perception donnée p, la probabilité que l'action a soit choisie est proportionnelle à la récompense w qu'il a reçue dans le passé, et inversement proportionnelle au nombre d'essais t. Cette pondération assure l'exploration de tous les comportements possibles. L'ensemble des règles pondérées ainsi défini est stocké dans une matrice d'action comme celle présentée en figure 4.5 (sur la figure n'apparaissent que les poids de déclenchement). Ce type de matrice a déjà été utilisé dans des travaux précédents pour permettre à un robot de choisir la meilleure action dans une situation donnée [Chapelle, 2001].

Découpeur

			0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
T	n_SatP		000	001	002	010	011	012	020	021	022	100	101	102	110	111	112	120	121	122	200	201	202	210	211	212	220	221	222
0	3	Chercher	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
1	1	Pousser																											
2	2	Decouper			0,5			0,5			0,5			0,5			0,5			0,5			0,5			0,5			0,5
3	3	Approcher		0,5			0,5			0,5			0,5			0,5			0,5			0,5			0,5			0,5	
4	2(*)	Maintenir			0,5			0,5			0,5			0,5			0,5			0,5			0,5			0,5			0,5
5	3	Altruisme				0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6

FIG. 4.5 – Poids de déclenchement contenus dans une matrice d'action d'un robot découpeur en début d'expérience. Chaque ligne correspond à une action (exemple : chercher), chaque colonne une perception. Par exemple, dans la situation n°10, l'action *Approcher* aura plus de chance d'être déclenchée que l'action *chercher*, mais restera moins prioritaire que l'action *Altruisme*.

Ces règles sont donc utilisées dans un processus de sélection d'action qui s'inspire du modèle EMF introduit par [Drogoul, 1993]. La Figure 4.6 résume l'architecture et le principe de renforcement.

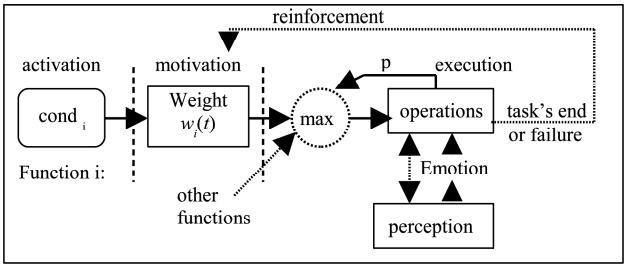

FIG. 4.6 – Représentation du comportement du robot et du processus de sélection et de renforcement.

Processus de sélection d'actions

Lorsque le robot reconnaît une situation s grâce à ses perceptions, il doit déterminer l'action à déclencher. Notre méthode consiste à faire un tirage aléatoire des actions tout en favorisant les actions gratifiantes (c'est-à-dire avec une forte récompense attendue) et en pondérant par le nombre d'essais. Ceci permet d'explorer toutes les actions possibles. En particulier celles où l'agent a peu d'expérience, et où il est peu sûr de la qualité de la valeur de la force de déclenchement. Nous donnons ci-après l'algorithme de sélection d'action.

$MAct$: **matrice d'action**

Pour toutes les actions i **faire**
  Si (l'action i n'est pas inhibée) **Alors**
    Calculer le poids de i :
    $\text{Poids}(i) = \frac{\text{Valeur}(\text{MAct}[i][s])+1}{\text{Nb expériences}(\text{MAct}[i][s])+1}$
    Retenir l'action et son poids
  Fin Si
Fin Pour

Calculer la somme des poids des actions :

$$S = \sum_j \text{Poids}(j)$$

Calculer *Choix* :

$$\text{Choix} = \text{random} \times S$$

Choisir l'action e telle que :

$$\sum_j^{e-1} \text{Poids}(j) \leq \text{Choix} \leq \sum_j^{e} \text{Poids}(j)$$

ALGORITHME 1 – Algorithme de sélection d'action

Dans le calcul du poids de l'action i, on ajoute 1 à l'ancienne valeur de la matrice afin de permettre à toute action d'être essayée, même si la valeur initiale de celle-ci est nulle.

Renforcement après la fin d'une expérience

Pendant toute la durée de l'expérience, l'agent écoute la valeur de l'émotion à laquelle il est rattaché. La moyenne (notée $\overline{\text{emotion}_i}$) est calculée incrémentalement, en fonction de la valeur de emotion_i perçue, par :

$$\overline{\text{emotion}_{i_{t+1}}} = \frac{\overline{\text{emotion}_i} \times t + \text{emotion}_{i_t}}{t+1}$$

Une expérience se termine quand l'une des conditions suivantes est vérifiée :
- L'action n'est plus adaptée à la situation, l'action doit être inhibée.
- La durée maximum de l'expérience a été atteinte. Cette limitation dans le temps permet aux robots de multiplier leurs expériences.

De plus, l'expérience doit avoir une durée minimum pour pouvoir être prise en compte dans l'apprentissage. Si c'est le cas, on renforce la valeur correspondant à la situation s détectée en début d'expérience, pour l'action i en calculant :
- Δ qui est la différence entre la moyenne de l'émotion et de sa valeur initiale
- r est la récompense associée à l'expérience. On donne de l'importance à la variation Δ. Si Δ est grand, il est prépondérant, sinon on utilise la moyenne de l'émotion. En effet, si la valeur l'émotion est restée élevée (comme sur la figure 4.7), il est important de renforcer l'expérience, et Δ étant faible, la valeur la plus pertinente à utiliser pour le renforcement est la moyenne de l'émotion.
- β est un coefficient de renforcement, plus il est faible plus l'impact de la récompense de la dernière expérience sera important. La valeur de β utilisée dans nos expériences est de 0.9 (valeur classique pour les algorithmes d'apprentissage par renforcement).

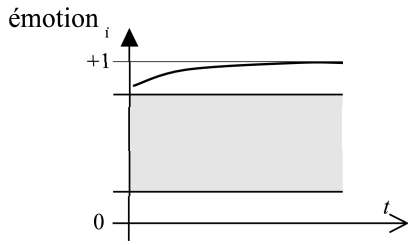

FIG. 4.7 – Cas où l'émotion est restée à une valeur élevée pendant toute la durée de l'expérience.

Le processus de renforcement est donc le suivant :

Soit n le nombre d'expériences de l'action i pour la situation s,
v la force de déclenchement de l'action i pour la situation s,

$\Delta = \overline{\text{emotion}_i} - \text{emotion}_{i_{t_0}}$

$r = \frac{|\Delta| \times \Delta + (1 - |\Delta|) \times \overline{\text{emotion}_i}}{2}$

Calcul de la nouvelle valeur de v : (renforcement)

$v' = \beta \cdot v + (1 - \beta) \cdot r$

Pour l'expérience (Situation s, action i) :
Stocker la nouvelle valeur v' dans la matrice
Incrémenter le nombre d'expérience n

Valeur(MAct[i][s]) = v'
Nombre d'expériences(MAct[i][s]) = $n + 1$

ALGORITHME 2 – Algorithme de renforcement

Nous venons de présenter la méthode d'apprentissage que nous avons choisie et le processus de sélection d'action qui permet de choisir la meilleure action dans une situation donnée. Mais nous n'avons pas précisé comment le robot identifiait une situation donnée en fonction de ses perceptions et comment le fait d'activer un comportement se traduisait par un changement d'état des effecteurs. En effet, dans nos précédents travaux [Chapelle *et al.* , 2002] nous avions écrit les fonctions qui permettaient d'identifier une situation en fonction des valeurs reçues des capteurs. Et de la même manière, nous avions écrit les différents comportements parmi lesquels le robot avait à choisir grâce à sa matrice d'actions.

Dans le problème qui nous intéresse nous souhaitons justement que le robot puisse construire lui même les structures qui lui permettront de percevoir le monde et d'agir sur celui-ci. Or les capteurs qui constituent le corps du robot ne donnent pas directement une perception mais une quantité de valeurs continues. Quant aux effecteurs, on ne peut se contenter de les activer : si l'on prend le cas d'un moteur électrique, sa vitesse de rotation varie en fonction de la quantité de courant qui lui est envoyée. Nous avons donc besoin de discrétiser les valeurs reçues en entrée par les capteurs, afin de distinguer un nombre fini d'action exécutables.

4.1.3 Discrétisation des valeurs : une approche basée sur la logique floue

Les algorithmes d'apprentissage par renforcement gèrent des règles dans lesquelles pour une perception donnée p, une action a est déclenchée avec une probabilité w. Ce modèle est bien adapté pour sélectionner un comportement plutôt qu'un autre, mais il ne peut s'appliquer à un modèle où les perceptions et les actions sont exprimés avec des valeurs continues (dans notre modèle, ces valeurs appartiennent à $[0\,;1]$). Chaque perception ou action peut prendre une infinité de valeurs. Prendre chacune de ces valeurs pour construire des règles amènerait à un nombre infini de règles.

Pour éviter cette explosion combinatoire, nous choisissons de regrouper les valeurs en intervalles, et pour bénéficier de nuances nous avons choisi de nous inspirer de la théorie des sous-ensembles flous [Zadeh, 1965]. L'objectif principal de cette théorie des ensembles flous est de permettre l'utilisation de la notion de *nuance* qui existe entre les perceptions, les actions ou les concepts qui sont manipulés. En effet, la logique booléenne ne permet de différencier que deux états (tout ou rien, vrai ou faux, "1" ou "0", près ou éloigné, etc.). La logique floue, quant à elle, permet de caractériser un état de façon graduelle. Ainsi, un état est représenté par son degré d'appartenance à un sous-ensemble flou donné, en utilisant la fonction d'appartenance définie pour cet ensemble, notée $\mu(x)$ sur les figure 4.8 et 4.10. L'utilisation des concepts liés à cette théorie permet alors de ne manipuler plus que deux états uniques. Par exemple, pour caractériser le degré de confort dans un système de climatisation, la logique booléenne ne permet de définir que deux états "chaud" et "froid", comme le montre la figure 4.8. Avec la logique floue, le degré de confort pourra être caractérisé en utilisant le nombre de nuances de température nécessaire pour gérer le comportement du système de climatisation.

Ce découpage en sous-ensembles flou permet donc de discrétiser une valeur continue en entrée du système de contrôle, ou alors de donner une grandeur réelle à partir de la commande générée par le contrôleur. La figure 4.9 montre les différentes parties d'un tel contrôleur, appelé *contrôleur flou*, et pour lesquelles nous donnons ici une courte description :

- la *fuzzification* : cette phase est chargée de convertir des grandeurs réelles en variables comprises par le contrôleur, nommées *variables linguistiques*.
- la *base de connaissance*, généralement écrite par un expert, caractérise la forme, et la position des fonctions d'appartenance.
- les *règles floues* permettent de faire la relation entre les entrées et les sorties sous la forme de règles *Si EntreeA Alors SortieB*. Afin de permettre de manipuler simultanément plusieurs entrées (et/ou sorties), on peut adjoindre à ces règles d'autres opérateurs logiques : OU, ET, etc.

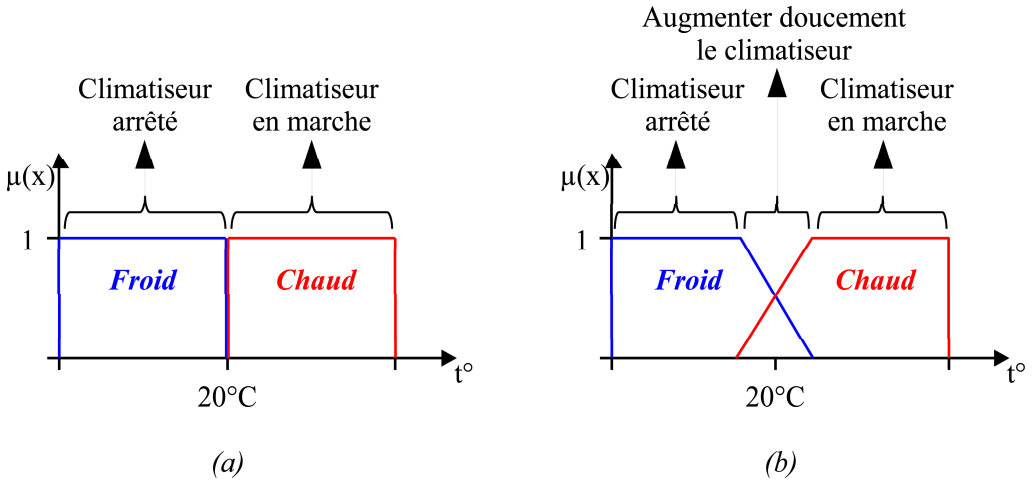

FIG. 4.8 – Exemple d'utilisation pour un système de climatisation. (a) Avec la logique booléenne il fait soit chaud, soit froid, ce qui pose des problèmes d'oscillation entre ces deux états. (b) Avec un découpage en sous-ensembles flous, on possède une phase de transition entre les états « chaud » et « froid » que l'on pourra associer à une action plus nuancée que « climatiseur arrêté » ou « climatiseur en marche ».

– la *défuzzification* fait alors la conversion de la commande générée par le contrôleur en une grandeur réelle appliquée sur les actionneurs.

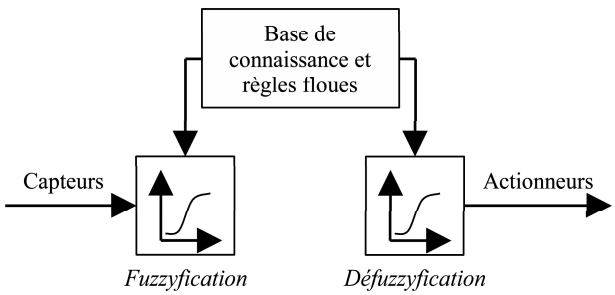

FIG. 4.9 – Exemple d'un contrôleur flou.

Nous avons donc décidé de ne considérer qu'un nombre fini d'états donnés par une partition floue forte[1] de l'espace des états. Au lieu d'un ensemble infini de valeurs, une perception (ou une action) peut être exprimée en utilisant des états comme « peu », « moyen » ou « beaucoup ». La correspondance entre l'espace de valeurs continues et l'ensemble d'états finis est faite en utilisant une fonction de transfert telle que celle présentée sur la figure 4.10.

Grâce à ce genre de discrétisation des perceptions et des actions, l'utilisation d'une méthode d'apprentissage par renforcement devient possible car il y a un nombre fini

[1] partitionnement pour lequel la somme des degrés d'appartenance égale 1

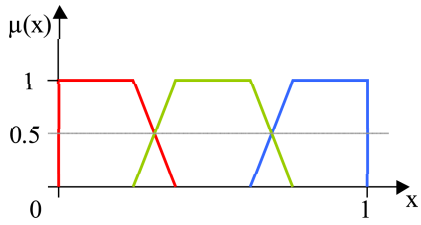

FIG. 4.10 – Exemple d'une fonction de transfert avec ses trois sous-ensembles flous associés.

de perceptions et d'actions. De plus l'utilisation de telles fonctions de transfert permet pour chaque perception, ou action, d'apprendre quels sont les paramètres les mieux représentatifs de son mode de fonctionnement :

- la *fonction de transfert appropriée* : linéaire, logarithmique ou avec un certain seuil (figure 4.11).
- le *nombre d'états différents* qui sont suffisants pour représenter les valeurs les plus intéressantes.

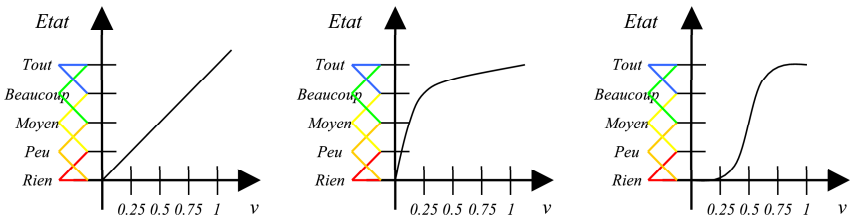

FIG. 4.11 – Exemples de fonctions de transfert. De gauche à droite : linéaire, logarithmique, avec seuillage.

4.2 Une architecture multi-agents

Les propriétés d'adaptation et d'évolution qui sont nécessaires à la réalisation d'une structure de contrôle pour nos robots nécessitent une architecture qui soit capable d'auto-organisation et qui permette l'émergence de comportements intelligents. De plus, la structure de contrôle et les besoins d'apprentissage requièrent la simulation d'un grand nombre d'entités apprenantes, qui ont des buts et des comportements différents, et qui doivent travailler ensemble dans le même environnement.

De tels besoins nous ont amené à considérer les systèmes multi-agents qui sont bien adaptés à la simulation d'un grand nombre d'entités apprenantes autonomes, et de leurs interactions. En outre, la programmation orientée agent présente des caractéristiques adaptées à ces objectifs :

- Les agents collaborent et se coordonnent afin de produire des performances qualitativement supérieures à celles des unités. De leurs interactions entre eux, émergent des comportements complexes.
- Les agents sont autonomes dans leur processus décisionnel (un agent décide par lui-même de l'action à exécuter).
- Des groupes d'agents peuvent être créés, au sein desquels les agents peuvent se rencontrer, partager et échanger des informations. Un groupe est un espace d'interaction où les agents ont une perception locale de l'environnement dans lequel ils vivent, les communications n'étant possibles qu'entre membres du même groupe.
- Les agents communiquent par échange de messages.

Les caractéristiques que l'on souhaite donner aux entités apprenantes sont proches de celles listées ci-dessus :

- La collaboration et la coordination des colonnes entre elles permettent de produire des performances qualitativement supérieures à celles de chaque colonne : c'est l'intelligence collective. De leurs interactions entre elles, émergent une structure de contrôle et des comportements complexes.
- Chaque colonne est autonome : elle peut choisir de filtrer les signaux qu'elle reçoit sur ses entrées et la modulation qu'elle va appliquer à ces informations.
- Les colonnes se rassemblent dans des groupes appelés *aires*.
- Chaque colonne a une perception locale de l'information. En effet, même si une colonne peut avoir beaucoup de connections, elle ne peut avoir une vue complète du système. De plus, une colonne peut envoyer ou recevoir des informations vers ou depuis d'autres colonnes qui sont dans son voisinage, comme des messages échangés entre agents d'un même groupe.

En utilisant la programmation par systèmes multi-agents, nous pouvons ainsi bénéficier des propriétés d'auto-organisation et d'émergence des systèmes multi-agents pour simuler le processus d'apprentissage réalisé au sein des colonnes et des aires corticales.

Nous allons donc présenter la structure de contrôle que nous proposons et son fonctionnement. Pour cela nous détaillerons chaque agent impliqué dans cette organisation et leurs rôles respectifs.

4.2.1 Les groupes d'agents : des aires

Pour chaque type de perception, il existe une aire corticale sensorielle et pour chaque type d'effecteur, une aire motrice. Une aire associative fait alors la relation entre chaque paire d'aire motrice et sensorielle. Enfin, l'aire frontale contrôle toutes les aires associatives. C'est cette dernière qui reçoit les émotions / motivations et qui doit coordonner

l'activité des aires associatives (l'apprentissage et la sélection d'actions) en diffusant ou non les émotions perçues.

Une aire d'apprentissage est un espace d'interaction privilégié qui remplit une certaine fonction, et au sein de laquelle les colonnes échangent des informations. De la même manière, un groupe d'agents est un espace d'interaction privilégié qui réalise un comportement, et au sein duquel les agents interagissent par envoi de messages.

Nous allons donc voir plus précisément dans les sous-sections qui suivent quels sont les agents et les groupes que nous implémentons pour réaliser notre structure de contrôle en nous inspirant, respectivement, des colonnes et des aires corticales. Cette structure de contrôle aura pour but de réaliser la connexion entre les entrées, les sorties et les émotions (voir illustration en figure 4.12). Nous commencerons donc par présenter le modèle général d'un de nos agent. Puis nous verrons les agents qui sont les plus spécifiques : ceux qui se situent aux frontières de la structure de contrôle, à savoir les agents appartenant à la couche physique et à la couche émotionnelle. Nous nous intéresserons ensuite au contenu proprement dit de la structure de contrôle constitué par des agents plus génériques chargés de mettre en relation émotions, entrées et sorties. Enfin, nous présenterons le fonctionnement de la structure de contrôle ainsi créée, ainsi qu'un exemple d'application d'une telle réalisation.

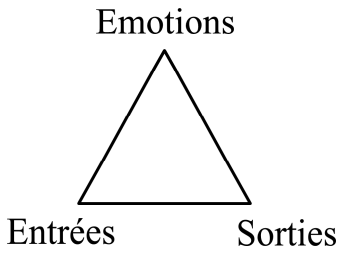

FIG. 4.12 – La structure de contrôle se situe entre des entrées, des sorties et des émotions. Le triangle symbolise la limite entre la structure de contrôle (à l'intérieur) et les contraintes liées à l'environnement et au concepteur (à l'extérieur).

4.2.2 Modèle général d'un agent

En sous-section 4.2 nous avons proposé un parallèle entre agent et colonne corticale. En effet nous choisissons de modéliser nos agents dans le but, non pas de modéliser fidèlement des colonnes corticales, mais plutôt de reproduire certains principes de fonctionnement qui nous semblent importants. Dans cette section nous allons donc présenter la modélisation que nous avons retenu pour constituer la structure de contrôle dont nous avons besoin.

Les agents qui constituent la structure de contrôle du robot sont issus du même modèle que nous allons présenter ici. La structure interne de l'agent est illustrée par la figure 4.13. Chaque agent utilise une fonction de couplage f pour mettre en relation un ensemble d'entrées E avec un ensemble de sorties S. Cette fonction de couplage est soumise à une émotion Em qui renforce le poids de déclenchement d'une sortie s_i pour une entrée donnée e_j selon un algorithme de renforcement tel que celui décrit en sous-section 4.1.2. La fonction de couplage f permet à l'agent de choisir l'action j qui a donné, par le passé, les récompenses les plus élevées (calculées en fonction de l'évolution de l'émotion Em), et qui a donc le plus de chance de donner à nouveau des récompenses positives. Il est donc capable de donner en retour la récompense maximale attendue notée Rew.

Cette fonction de couplage peut prendre diverses formes, la seule restriction est qu'elle doit pouvoir modifier son comportement en fonction de l'émotion perçue et être également capable de donner une estimation de la récompense attendue dans une situation donnée. Si l'on souhaite, par exemple, obtenir la perception d'un objet particulier à partir d'un flux vidéo, on pourrait mettre comme fonction de couplage un réseau de neurones artificiel.

En ce qui nous concerne, nous allons utiliser comme fonction de couplage une matrice d'action telle que celle présentée sur la figure 4.5 dans la sous-section 4.1.2 qui traite d'apprentissage par renforcement. Dans ce cas la récompense maximale attendue Rew est donnée par la matrice d'action en sélectionnant le poids de déclenchement le plus élevé parmi les actions déclenchables pour l'entrée e_j. Cette valeur que produit un agent n'est pas utilisée dans nos travaux car les autres informations produites par les différents agents suffisent à réaliser une structure de contrôle opérationnelle et apprenante. C'est pourquoi dans les figures et les explications qui suivent nous ne ferons plus référence à cette valeur Rew.

Les entrées E et les sorties S de l'agent peuvent être reliées à des capteurs ou des moteurs appartenant à la structure physique du robot (représentés sur le bas de la figure 4.13). Mais nous verons plus tard, lorsque nous présenterons les *représentants* en sous-section 4.2.5, que les entrées ou les sorties d'un agent peuvent être connectées à d'autres agents de la structure de contrôle. En effet, nous verrons qu'un agent peut recevoir des **D**emandes d'**A**ction de la part d'un autre agent, cela est représenté par la flèche *Demande d'Action* représentée en pointillés sur la figure 4.13. De la même manière un agent peut, à partir de ses entrées, produire une information perceptive qui pourra être utilisée par d'autres agents et que nous avons représenté par la flèche en pointillés *Perception*. Il est important d'insister sur le fait que les termes *Demande d'Action* et *Information Perceptive* employés sur cette figure désignent respectivement des actions que l'on pourra demander à cet agent d'effectuer et des perceptions que cet agent produira à destinations d'autres agents.

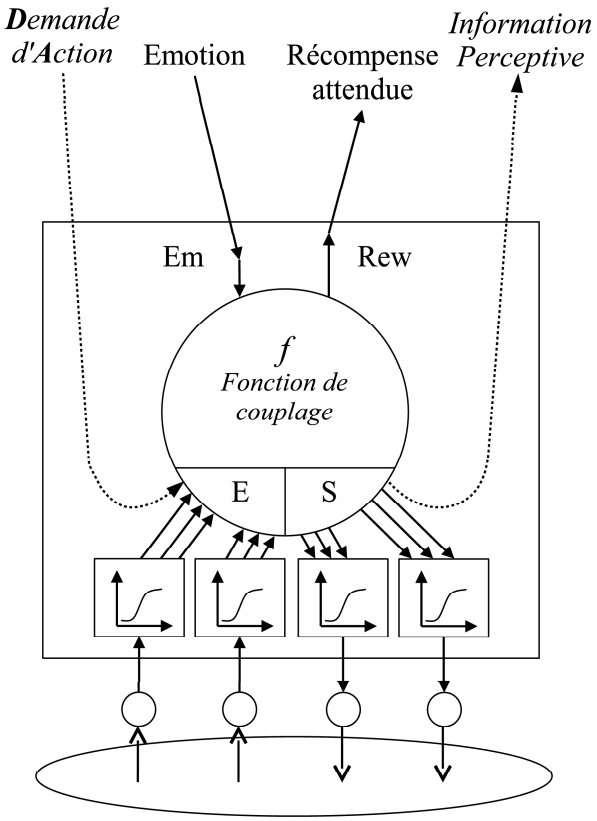

FIG. 4.13 – Modèle général d'un agent.

4.2.3 La couche physique

La couche physique joue le rôle d'interface entre le corps du robot, constitué de divers effecteurs et capteurs, et l'esprit constitué d'entités virtuelles. Cette couche physique, illustrée par la figure 4.14, permet donc à l'architecture de contrôle de percevoir l'environnement et d'agir sur celui-ci. On peut faire de cette décomposition une analogie avec un système de téléopération permettant à un individu de piloter un robot à distance comme le montre la figure 4.15.

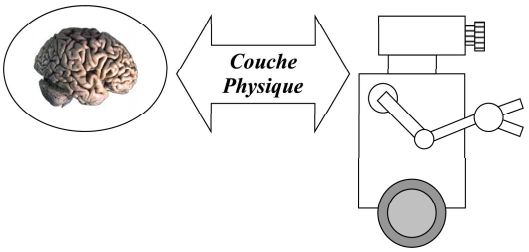

FIG. 4.14 – Les agents de la couche physique réalisent l'interface entre l'esprit, représenté à gauche de la figure, et le corps, représenté à droite.

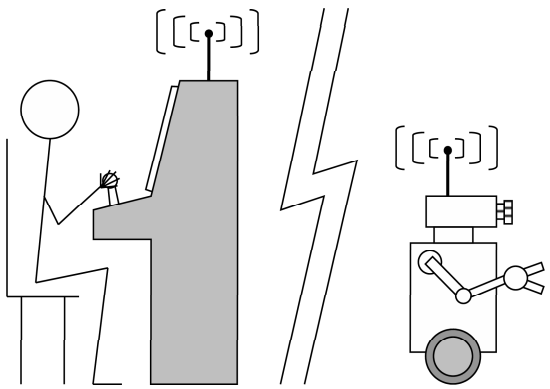

FIG. 4.15 – Robot piloté à distance. L'esprit est représenté par l'opérateur humain. Celui-ci peut voir ce que les caméras du robot perçoivent et peut piloter les effecteurs du robot au moyen du pupitre de commande représenté au centre. Ce pupitre s'apparente à la couche physique dont nous parlons.

Les agents appartenant à cette couche sont directement reliés aux capteurs et aux effecteurs du robot. Ils proposent donc au système les moyens d'interagir avec le monde extérieur comme le montre la figure 4.16.

Au sein des agents appartenant au groupe de la couche physique, on distingue alors deux types d'agents, selon qu'ils font entrer de l'information dans le système ou en sortir :

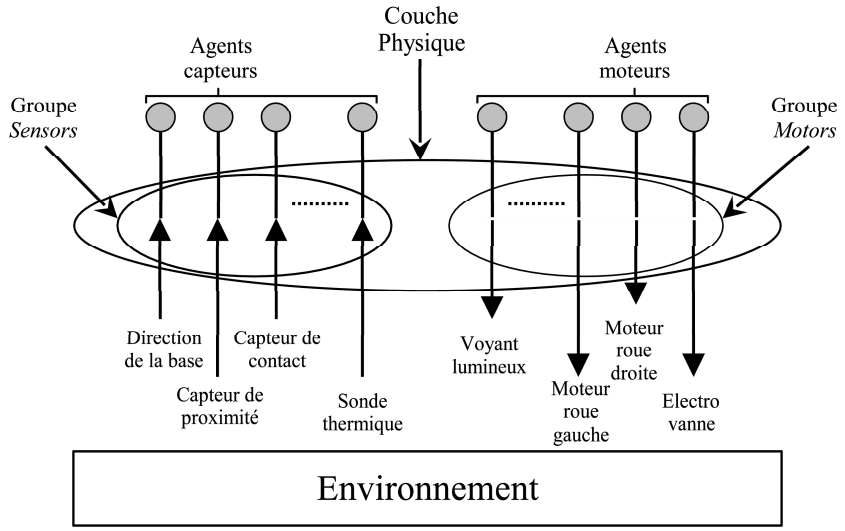

FIG. 4.16 – La couche physique regroupe les agents capteurs et les agents moteurs.

Les agents capteurs : les entrées

Les agents capteurs sont les agents qui font entrer de l'information dans le cerveau. Ceux-ci appartiennent à un même groupe appelé *Sensors*. Chaque agent appartenant à ce groupe produit une valeur dans l'intervalle $[0, 1]$ qui est directement fonction de l'état du capteur associé.

Par exemple un capteur de contact fournira un 0 lorsqu'il est au repos, et un 1 lors qu'il est en contact avec un objet. Un capteur de proximité fournira, quant à lui, une valeur continue proportionnelle à la distance mesurée avec un objet.

Les agents moteurs : les sorties

Les agents moteurs sont les agents qui font sortir l'information du cerveau. Ils appartiennent au même groupe appelé *Motors*. Chaque agent appartenant à ce groupe peut recevoir une valeur comprise entre 0 et 1 qu'il répercute sur l'actionneur auquel il est associé. Comme pour les agents capteurs, le moteur piloté par un agent capteur peut reconnaître deux états (allumé, éteint) ou bien une valeur continue (comme une vitesse de rotation).

Les figures 4.17(a) et 4.17(b) montrent la structure d'un agent capteur et d'un agent moteur. On peut remarquer que par rapport au modèle d'agent donné en sous-section 4.2.2 de nombreux points ont été fixés. Pour commencer, la fonction de couplage f a pour rôle de normaliser les valeurs qui entrent ou sortent du système en ramenant celles-ci dans l'intervalle $[0, 1]$. C'est pourquoi les agents de la couche physique ne sont reliés à aucune émotion : la fonction qui réalise la normalisation des valeurs est fixée

lors de la création d'un agent de la couche physique et n'évoluera pas par la suite. Enfin, on peut noter qu'il n'y a pas de processus de fuzzyfication ou de défuzyfication, en effet on laisse aux agents qui vont se greffer sur ces agents de la couche physique, le soin d'effectuer leur propre découpage en sous-ensembles de l'espace de variation des valeurs du capteur ou du moteur.

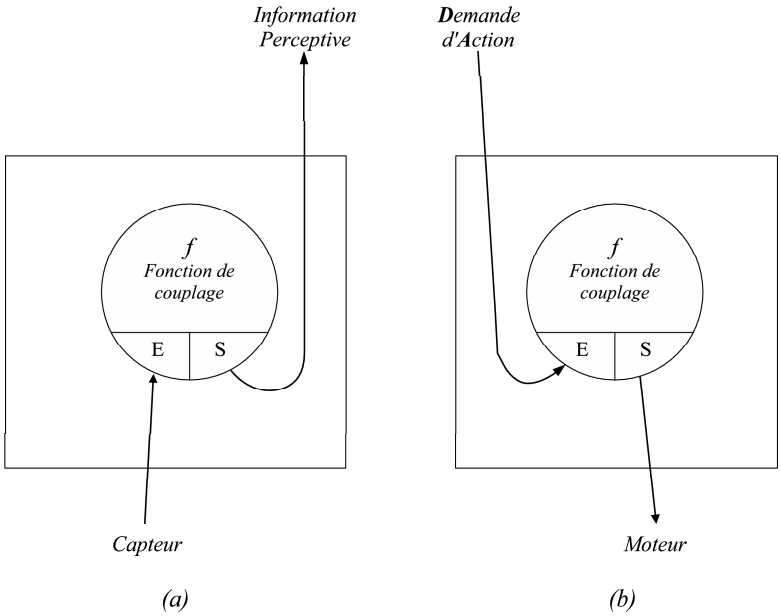

FIG. 4.17 – Les agents de la couche physique. A gauche (a) un agent capteur, à droite (b) un agent moteur.

4.2.4 La couche émotionnelle

La couche émotionnelle est la couche la plus éloignée de la couche physique présentée précédemment. C'est cette couche qui guide le comportement du robot et lui permet de sélectionner les actions appropriées à la satisfaction des émotions. Celle-ci est matérialisée par un groupe d'agents que nous appellerons *couche émotionnelle* représenté figure 4.18. Ce groupe est constitué d'un agent appelé *gestionnaire d'émotions* et d'autant d'agents appelés *délégués aux émotions* qu'il existe d'émotions.

Le gestionnaire d'émotions

Le gestionnaire d'émotions est l'agent qui reçoit les émotions que nous avons présentées en sous-section 4.1.1. Chaque émotion est directement transmise à un agent délégué qui en est responsable. Le modèle d'un tel agent est présenté en figure 4.19.

Le gestionnaire d'émotions va, quant à lui, choisir l'émotion qu'il souhaite satisfaire, et demandera au délégué concerné de trouver une solution. Pour effectuer cette sélection,

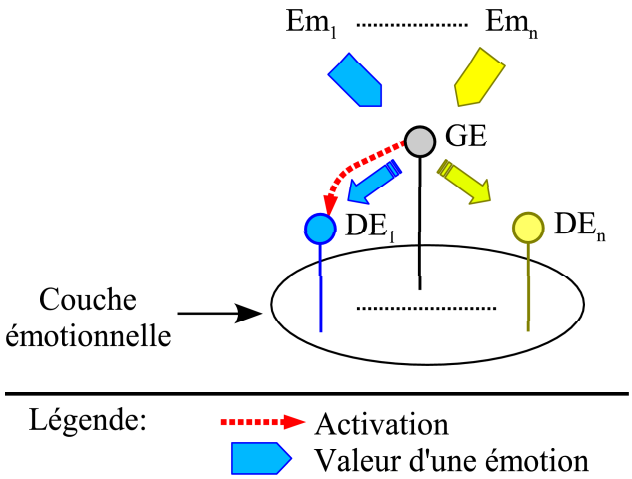

FIG. 4.18 – La couche émotionnelle regroupe le **G**estionnaire d'**É**motions, en gris au centre de la figure, et les **D**élégués aux **É**motions. Le gestionnaire d'émotions reçoit, pour chaque émotion, une valeur qu'il transmet au délégué concerné. Puis il choisis quel besoin satisfaire en activer le délégué correspondant.

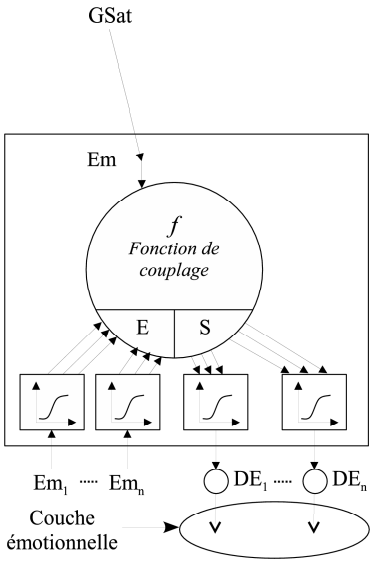

FIG. 4.19 – Modèle du gestionnaire d'émotions. Les perceptions de cet agent sont les valeurs Em_i de chaque émotion. Celui-ci agit en activant un délégué à une émotion DE_j. Les règles de déclenchement contenue dans f sont renforcées par $GSat$, l'état émotionnel du robot (définit en sous-section 4.1.1).

le gestionnaire d'émotions possède une matrice comme celle présentée figure 4.5, qui définit pour chaque situation émotionnelle $Sit_{i\in I}$ et pour chaque délégué à une émotion $DE_{j\in J}$, un poids de déclenchement $w_{(i,j)}$ ainsi que le nombre de fois $t_{(i,j)}$ où ce délégué a été sollicité. Chaque situation émotionnelle représente une perception, et chaque délégué à une émotion représente une action possible.

Les couples (Perception,Action) forment donc un ensemble de règles de déclenchement qui sont définies par :

$$(Sit_{i\in I}, DE_{j\in J}) \rightarrow (w_{(i,j)}, t_{(i,j)})$$

Pour une situation émotionnelle $Sit_{i\in I}$, la probabilité de solliciter l'agent $DE_{j\in J}$ est proportionnelle à l'état émotionnel $w_{(i,j)}$ attendu, et inversement proportionnelle au nombre d'essais $t_{(i,j)}$. Le fait de prendre en compte le nombre d'essais pour chaque action permet d'essayer toutes les actions possibles, même si l'une d'elles n'a pas donné la récompense attendue lors d'un premier essai. J définit le nombre de délégués aux émotions que connaît le gestionnaire d'émotions. L'ensemble des situations émotionnelles $Sit_{i\in I}$ est défini comme la combinaison des états émotionnels de chaque agent.

Soit NE_i le nombre d'état émotionnels du délégué à l'émotion i, le nombre J de situations émotionnelles est donné par :

$$J = \prod_{i\in I} NE_i$$

Considérons que notre système possède 4 émotions (comme sur la figure 4.18), et que pour chaque émotion on distingue 3 états émotionnels différents donnés par un découpage en 3 sous-ensembles flous de l'espace de variation de chaque émotion (voir processus de *fuzzification* de la sous-section 4.1.3). Dans cet exemple, $I = 4$, et $J = 3^4 = 81$. Le système reconnaîtra donc 81 situations émotionnelles différentes, et disposera donc de 324 (81×4) règles de déclenchement.

Les délégués aux émotions

Chaque délégué à une émotion est un agent qui *connaît* d'autres agents[2] lui permettant de satisfaire l'émotion à laquelle il est rattaché. Parmi ces agents que le délégué connaît, certains lui fournissent de l'information et peuvent être considérés comme des perceptions, et d'autres reçoivent de l'information et sont donc considérés comme des actions possibles. Le délégué dispose ainsi d'un ensemble de solutions (actions) parmi

[2]Nous verrons en sous-section 4.2.5 quels sont ces agents que le délégué *connaît*, et que l'on appelle *les représentants*.

lesquelles il peut choisir, pour une situation (perception) donnée, celle qui lui permet de répondre à son besoin.

Pour effectuer son choix, il dispose donc, comme pour le gestionnaire d'émotions, d'un ensemble de règles de déclenchement, défini par :

$$(\text{Perception}_{i\in I}, \text{Action}_{j\in J}) \rightarrow (w_{(i,j)}, t_{(i,j)})$$

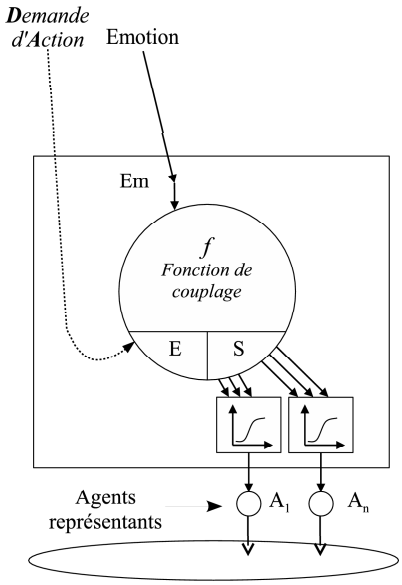

FIG. 4.20 – Modèle d'un délégué à une émotion. Cet agent perçoit demandes d'action provenant du gestionnaire d'émotions et agit en activant les agents qu'il resprésente. Son processus de sélection d'action exprimé par f est renforcé par l'émotion à laquelle il est rattaché.

4.2.5 Les représentants

Les représentants sont les agents qui s'interfacent et jouent le rôle d'intermédiaires entre la couche physique, dite *couche basse*, et la couche émotionnelle, dite *couche haute*. Un tel agent réalise donc une abstraction des agents inférieurs qu'il représente afin de produire une synthèse qui pourra être utilisée par les agents des couches supérieures. Un représentant permet donc de modéliser un concept abstrait par rapport aux agents qu'il représente.

Comment un tel agent peut il représenter d'autres agents ? Prenons l'exemple d'un robot muni de deux roues tel que le *Type 1* présenté en sous-section 6.1.2. Pour réaliser un déplacement tel que ceux présentés sur la figure 4.22, il doit activer ses deux roues en simultané. Or, pour chaque tâche que va exécuter le robot, on ne veut pas avoir

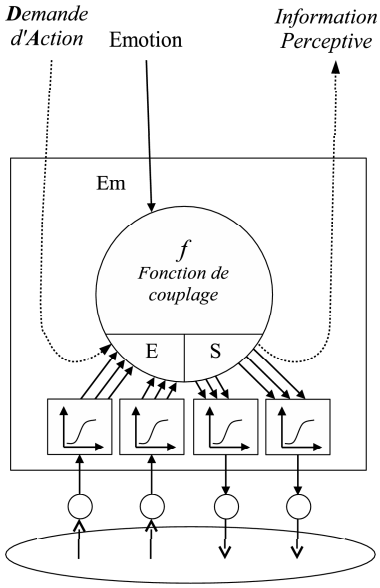

FIG. 4.21 – Modèle d'un représentant. Cet agent utilise une fonction de couplage f pour mettre en relation un ensemble d'entrées E avec un ensemble de sorties S. Cette fonction de couplage est soumise à une émotion Em qui renforce le poids de déclenchement d'une sortie s_i pour une entrée donnée e_j selon un algorithme de renforcement tel que celui décrit en sous-section 4.1.2.

à réécrire les commandes moteur qui permettent de se déplacer. Au lieu de cela, on aimerait que chaque tâche nécessitant un déplacement puisse disposer de commandes telles que « avancer », « reculer », ou encore « tourner à droite ».

C'est pour cela que nous avons trouvé judicieux de créer un agent qui va piloter les deux agents moteurs et proposer au système plusieurs configurations motrices abstraites. L'organisation correspondante de ces agents est présentée figure 4.23, et la structure du représentant moteur illustrée figure 4.24.a. Le rôle de ce nouvel agent est donc de représenter les deux agents moteurs, c'est pourquoi nous appellerons ce type d'agent un *représentant.*

Dès lors on peut concevoir plusieurs types de représentants selon qu'ils permettent d'abstraire des actions ou des perceptions. Dans l'exemple que nous avons exposé dans le paragraphe précédent, le représentant proposait des abstractions d'agents moteurs sous forme d'actions. L'abstraction d'agent capteur permet, quant à elle, de proposer une information perceptive simplifiée par rapport au flux d'informations qui arrivent aux agents capteurs.

Considérons à nouveau le robot *Type 1* qui dispose de huit capteurs infrarouge pour détecter la proximité d'objets. Un agent représentant ces huit agents capteurs permet de donner des perceptions telles que « je suis dans un couloir », « je suis enfermé » ou encore « le champ est libre » à n'importe quelle tâche qui aurait besoin de ce type

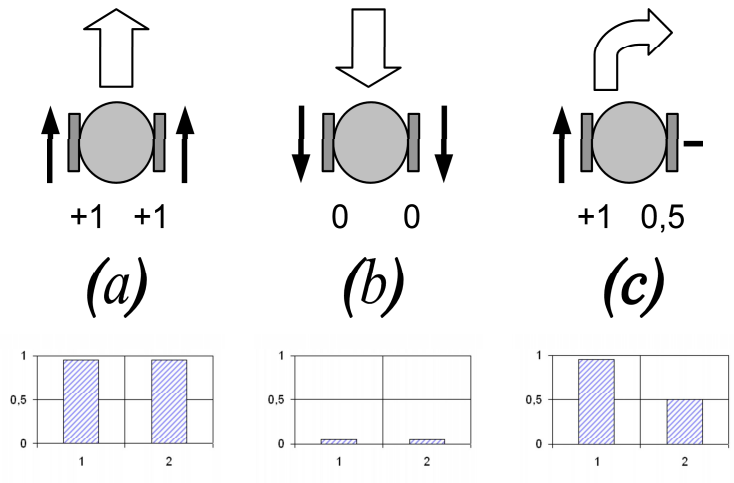

FIG. 4.22 – Exemple d'actions : (a) avancer, (b) reculer, (c) tourner à droite.

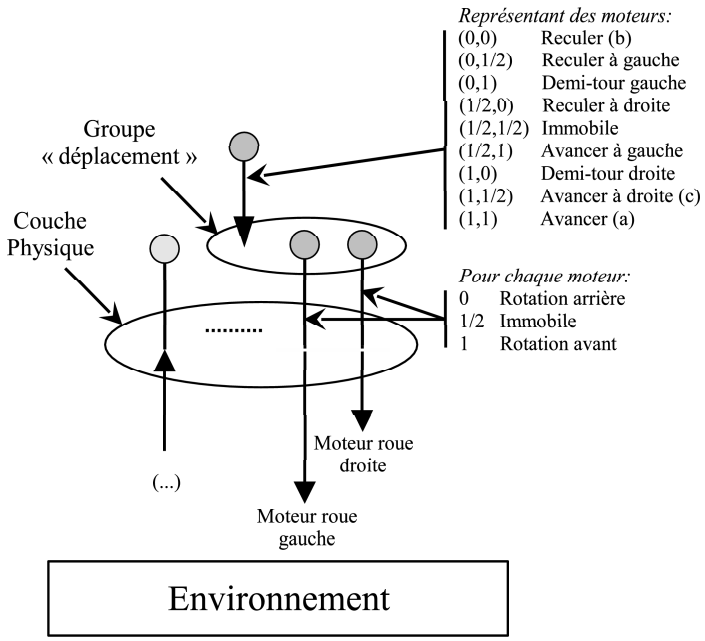

FIG. 4.23 – Organisation correspondant à la figure 4.22 sur laquelle on peut voir le représentant des moteurs et le groupe qu'il a créé pour communiquer avec chaque moteur. Pour chaque agent sont précisées toutes les actions possibles.

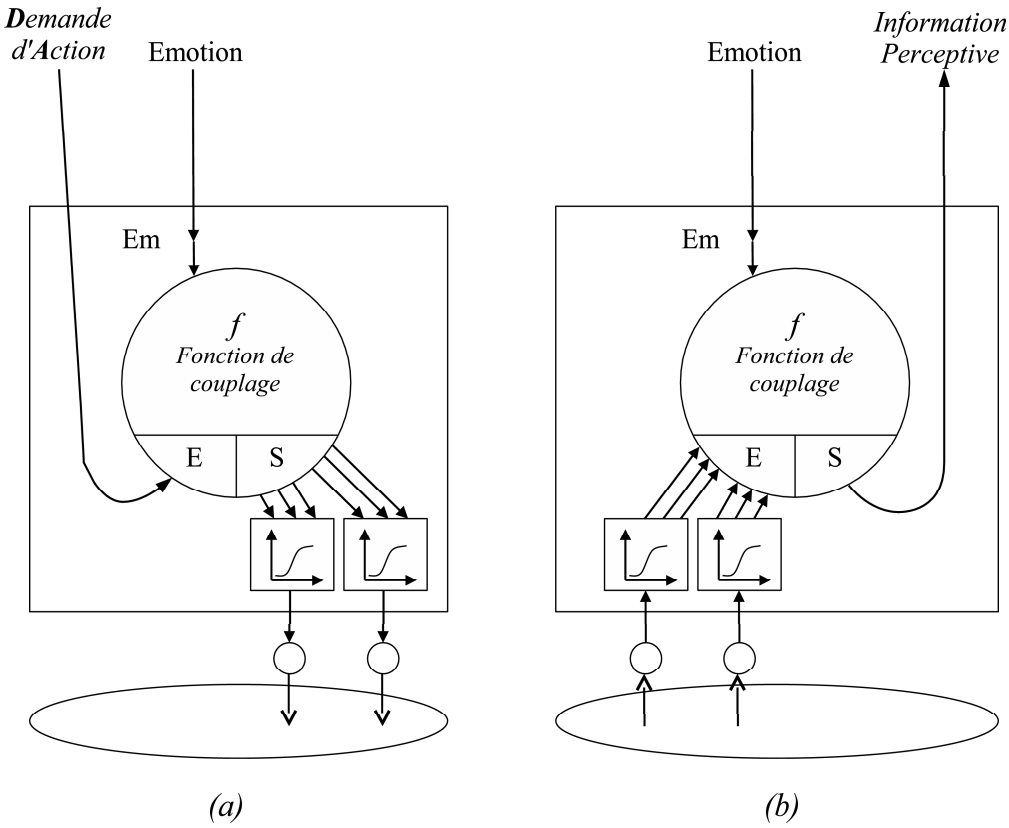

FIG. 4.24 – Illustration d'un représentant moteur (a) et d'un représentant capteur (b).

d'information. Ces trois perceptions sont présentées sur la partie supérieure de la figure 4.25. On peut également voir, pour chaque perception, les valeurs reçues des agents capteurs. L'organisation correspondante est présentée sur la figure 4.26, et la figure 4.24.b illustre la structure du représentant capteur.

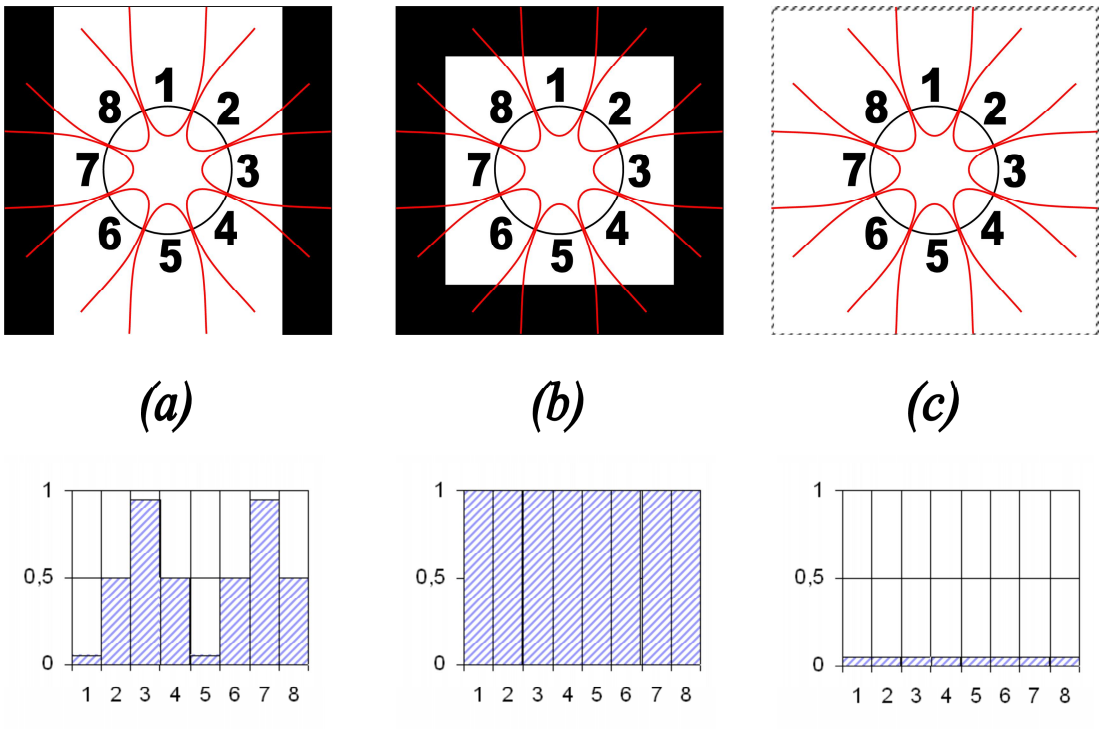

FIG. 4.25 – Exemple de perception : (a) couloir, (b) enfermé, (c) champ libre.

A cet instant, on pourrait penser qu'il existe deux types d'agents représentants, bien distincts : ceux qui proposent des actions, et qui représentent uniquement des agents moteurs, et ceux qui proposent des perceptions, et qui ne représentent que des agents capteurs. Ce serait oublier que certaines actions nécessitent des informations perceptives, et que certaines perceptions nécessitent de pouvoir agir sur le système ou l'environnement. Par exemple, l'action « se déplacer vers la base » illustrée par la figure 4.27, est une action qui nécessite bien sûr d'activer les moteurs, mais surtout de percevoir la direction de la base. De la même manière, lorsque le robot est contre un objet, la perception « objet déplaçable » est donnée par la perception simultanée d'une action sur les roues et de la perception d'un « non-déplacement ».

4.2.6 Le fonctionnement : un exemple

Pour montrer le fonctionnement des agents que nous venons de présenter, nous allons proposer une architecture de contrôle relativement simple permettant de piloter un robot tel que le *Type 1* présenté en sous-section 6.1.2.

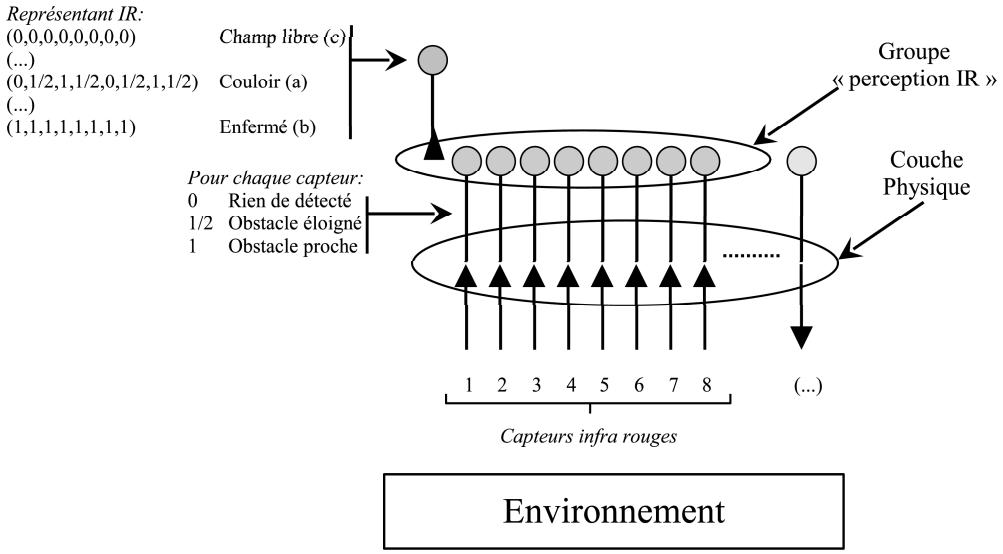

FIG. 4.26 – Organisation correspondant à la figure 4.25 sur laquelle on peut voir le représentant des capteurs infrarouges et le groupe qu'il a créé pour communiquer avec chaque capteur. Pour chaque agent sont précisées toutes les actions possibles.

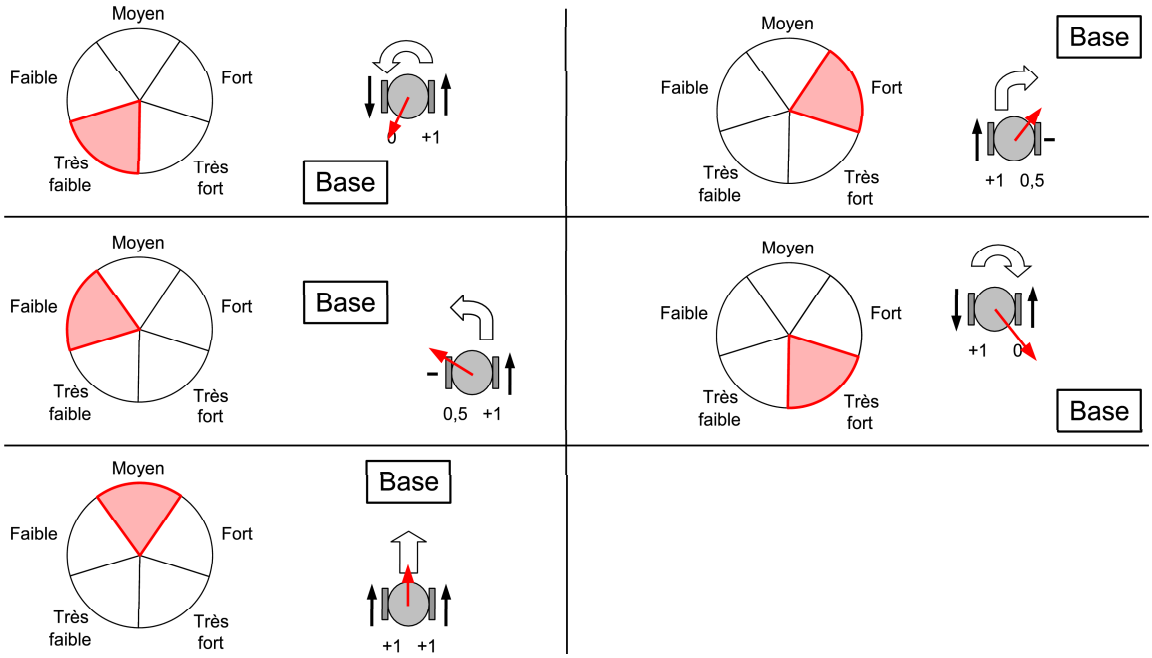

FIG. 4.27 – L'action « Se déplacer vers la base » nécessite de prendre en compte la perception « direction de la base ». Sur cette figure les termes « faible », « moyen », « fort » désignent les différents états donnés par un découpage en sous-ensembles flous de l'espace de variation du capteur de direction de la base.

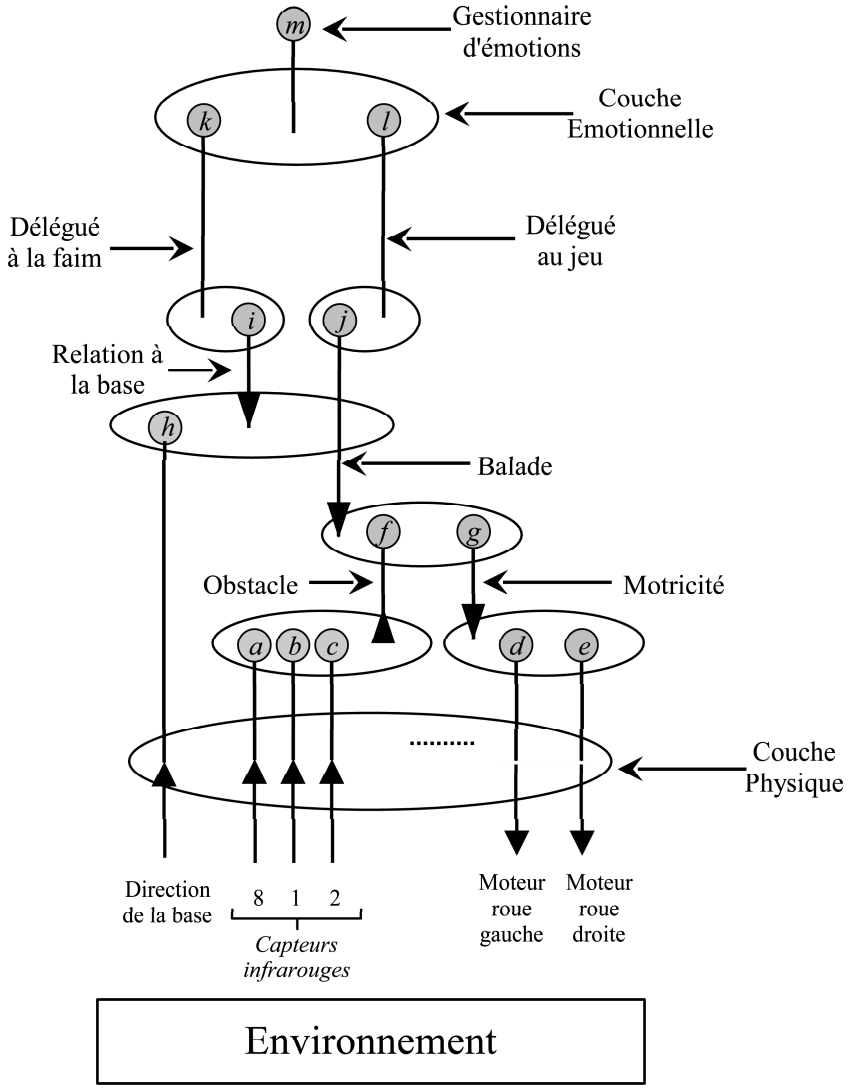

FIG. 4.28 – Exemple d'une architecture de contrôle permettant au *Type 1* de satisfaire les émotions de *Jeu* et de *Faim*.

Pour la réalisation de cette structure, présentée figure 4.28, nous allons ne considérer que deux émotions. La première, le *Jeu*, est liée aux déplacements qu'effectue le robot : plus il se déplace, plus l'émotion de *Jeu* est satisfaite. La seconde émotion, la *Faim*, est quant à elle liée à l'état de charge des batteries : plus les batteries sont chargées et plus cette émotion est satisfaite.

Le robot devra donc alternativement passer d'une tâche consistant à « jouer » à une tâche lui permettant de recharger ses accumulateurs. L'exécution de ces deux tâches ne nécessitant que peu de capteurs et d'actionneurs, nous ne considérerons qu'une partie de la couche physique disponible sur le *Type 1*, à savoir les trois capteurs infrarouges faciaux, le capteur de direction de la base et les deux moteurs permettant le déplacement.

Nous allons donc détailler le rôle et le fonctionnement de chacun des agents illustrés sur la figure 4.28 en commençant par les agents de la couche physique et en remontant jusqu'à la couche émotionnelle. Lorsque nous présenterons un agent, nous donnerons également une représentation condensée du contenu de la matrice d'action représentant la fonction de couplage, sous la forme d'un tableau tel que le tableau 4.1 présenté ci-dessous. Cette représentation sous la forme de règles de déclenchement est condensée car pour une perception donnée n'est inscrite dans le tableau que l'action au poids de déclenchement le plus élevé. Si la partie gauche de la règle est vrai (c'est à dire le couple (Demande d'Action,Perception)), l'action correspondante exprimée dans la partie droite est déclenchée. La colonne *DA* désigne les **D**emandes d'**A**ction que l'agent x reçoit de la part d'un autre agent y, quant à la colonne *Perception* elle désigne les informations perceptives émisent par les agents auquel l'agent x est connecté. Enfin, pour plus de compréhension, nous avons choisi d'indiquer dans la colonne *Légende* la sémantique qu'un humain pourrait donner à chacun des cas exprimés dans cette matrice.

TAB. 4.1 – Matrice d'action de l'agent x.

DA	Perception	Action	Légende
	distance $\in [0,1/3[$	$1/6$	Rien
	distance $\in [1/3,2/3[$	$3/6$	Proche
	distance $\in [2/3,1]$	$5/6$	Contact

Les agents capteurs infrarouges (a, b et c)

Comme nous l'avons précisé plus tôt, nous ne considérerons dans cet exemple que trois des huit capteurs infrarouges disponibles sur le robot et présentés sur la figure 4.25. En effet, les capteurs « Avant-gauche », « Avant » et « Avant-droit », respectivement étiquetés a, b et c sur la figure 4.28, sont suffisants pour permettre une navigation rudimentaire évitant les obstacles.

La valeur renvoyée par un capteur est une grandeur réelle fonction de la distance à l'obstacle perçu et nous souhaitons différencier les trois perceptions suivantes : « pas d'obstacle », « obstacle proche » et « contact ». C'est pourquoi l'espace de variation de chaque capteur est découpé en trois sous-ensembles comme le montre la figure 4.29.

Chaque sous-ensemble flou sera, par la suite, désigné par sa médiane. Ainsi le premier sous-ensemble est étiqueté $1/6$, le second $3/6$ et le dernier $5/6$ et ils correspondent respectivement aux perceptions « pas d'obstacle », « obstacle proche » et « contact ».

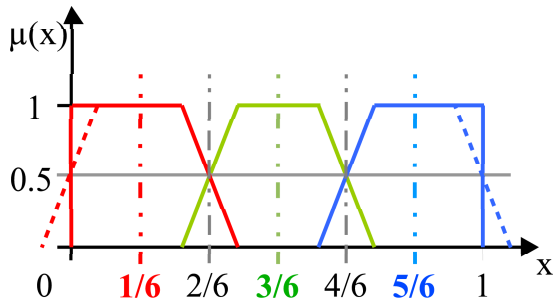

FIG. 4.29 – Découpage de l'espace de variation d'un capteur infrarouge en trois sous-ensembles flous. L'ordonnée de la médiane d'un sous-ensemble est utilisée pour nommer celui-ci, par exemple $1/6$ pour désigner le sous-ensemble de gauche correspondant à la perception « pas d'obstacle ».

Les matrices d'actions qui correspondent à chacun des trois agents sont présentées ci-dessous dans la table 4.2.

TAB. 4.2 – Matrice d'action de l'agent a (capteur infrarouge avant-gauche). Les matrices d'action des agents b et c (capteurs infrarouges avant et avant-droit) sont identiques.

DA	Perception	Action	Légende
	distance $\in [0, 1/3[$	$1/6$	Rien
	distance $\in [1/3, 2/3[$	$3/6$	Proche
	distance $\in [2/3, 1]$	$5/6$	Contact

Agent capteur de direction de la base (h)

Un autre type de capteur que nous avons choisi pour notre exemple est celui qui donne la direction de la base (l'agent h de la figure 4.28). On peut comparer ce capteur à une boussole dont l'aiguille serait attirée par un pôle nord magnétique sur lequel la base serait construite.

Afin de pouvoir obtenir une navigation plus précise, nous allons effectuer un découpage plus fin que celui que nous avons pris pour les capteurs infrarouges. En effet,

nous allons distinguer cinq perceptions différentes présentées en figure 4.30. La matrice d'action correspondante détenue par cet agent capteur est donnée par la table 4.3.

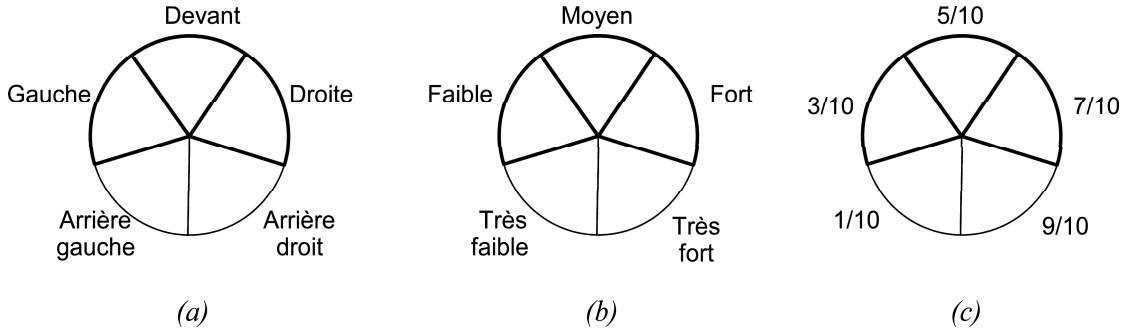

FIG. 4.30 – Le capteur de direction et ses cinq états possibles fonction du découpage de l'espace de variation du capteur en cinq sous-ensembles. A gauche (a) nommage topologique des sous-ensembles, au centre (b) nommage par rapport à la force du signal renvoyé par le capteur, et à droite (c) nommage par la valeur médiane de chaque sous-ensemble flou.

TAB. 4.3 – Matrice d'action de l'agent h (capteur de direction de la base).

DA	Perception	Action	Légende
		$^1/_{10}$	Arrière gauche
		$^3/_{10}$	Gauche
		$^5/_{10}$	Devant
		$^7/_{10}$	Droite
		$^9/_{10}$	Arrière droit

Agents moteurs roue gauche et roue droite (d et e)

A présent, nous allons nous intéresser aux deux agents moteurs qui entraînent les roues permettant au robot de se déplacer.

Chaque moteur peut avoir une vitesse variable et un sens de rotation en fonction de la valeur qui lui est envoyée comme illustré sur la figure 4.31. Les matrices d'action que nous avons choisies pour ces agents moteurs sont données par la table 4.4. Elles permettent de piloter les roues en choisissant parmi trois actions : « reculer », « être à l'arrêt » et « avancer ». Si l'on voulait obtenir des trajectoires et des déplacements plus fins, on aurait pu considérer cinq cas comme ceux présentés sur la figure 4.31.

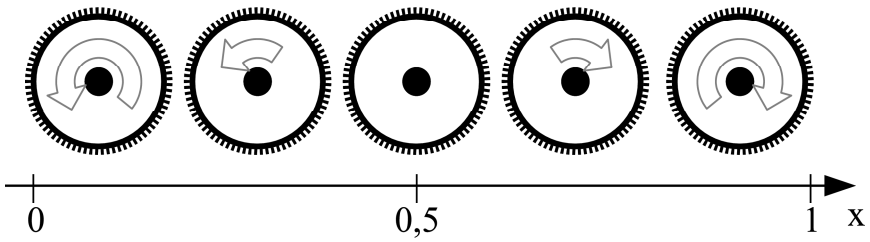

FIG. 4.31 – Sens de rotation d'un moteur en fonction de la valeur qui lui est envoyée. De 0 à $\frac{1}{2}$, le moteur tourne à l'envers. Pour une valeur de $\frac{1}{2}$, il est à l'arrêt, et au delà de $\frac{1}{2}$, il tourne à l'endroit.

TAB. 4.4 – Matrice d'action de l'agent d (moteur roue gauche). La matrice d'action de l'agent e (moteur roue droite) est identique à celle-ci.

DA	Perception	Action	Légende
		$\frac{1}{6}$	Marche arrière
		$\frac{3}{6}$	Immobile
		$\frac{5}{6}$	Marche avant

Représentant « Obstacle » (f)

A présent nous allons présenter un des agents qui suscitent chez nous plus d'intérêt : un *représentant.*

Le premier représentant que nous présentons est l'agent f qui représente les trois capteurs infrarouges (agents a, b et c), et va ainsi proposer un ensemble de perceptions qui seront utilisées par d'autres représentants. La matrice d'action de cet agent représentant est donnée par la table 4.5. Dans cette matrice nous avons choisi de n'inscrire, pour chaque perception, que l'action qui dispose du poids de déclenchement le plus élevé. Dans la pratique, et dans le cadre d'une structure figée non apprenante, on mettrait le poids de déclenchement à 1 pour l'action devant être exécutée pour une certaine perception. La matrice d'action donnée à ce représentant lui permet d'exprimer quatre perceptions différentes selon que le champ est libre ou qu'un obstacle est présent devant, à gauche ou à droite du robot.

Cela permettra donc à un autre représentant plus haut placé de choisir la meilleure trajectoire à effectuer en fonction des obstacles perçus.

Représentant « Motricité » (g)

A présent, regardons l'agent g, le représentant moteur. Celui-ci pilote les deux agents moteurs d et e, et propose donc plusieurs combinaisons motrices. Dans cet exemple,

TAB. 4.5 – Matrice d'action de l'agent f (Représentant « Obstacle »).

DA	Perception	Action	Légende
	(a=1/6,b=1/6,c=1/6)	1/8	Pas d'obstacle
	(a=1/6,b=3/6,c=5/6)	3/8	Obstacle devant droite
	(a=3/6,b=5/6,c=3/6)	5/8	Obstacle devant
	(a=5/6,b=3/6,c=1/6)	7/8	Obstacle devant gauche

nous ne considérons qu'une partie des différentes combinaisons motrices possibles. En effet, comme nous avons deux agents moteurs et que chacun de ses agents propose trois actions possibles, le nombre total de combinaisons s'élève à $3^2 = 9$. Nous avons donc choisi de ne conserver, parmi ces neuf combinaisons, que les cinq qui nous semblent les plus appropriées. Le représentant ainsi créé proposera donc aux autres agents les cinq actions possibles présentées dans la table 4.6.

TAB. 4.6 – Matrice d'action de l'agent g (Représentant « Motricité »).

DA	Perception	Action	Légende
1/10		(d=1/6,e=1/6)	Reculer
3/10		(d=3/6,e=5/6)	Tourner à gauche
5/10		(d=5/6,e=5/6)	Avancer
7/10		(d=5/6,e=3/6)	Tourner à droite
9/10		(d=3/6,e=3/6)	Immobile

Les agents qui feront appel à ce représentant pourront ainsi exécuter des tâches de déplacement sans avoir à se soucier de la combinaison adéquate à appliquer aux moteurs. Nous allons donc présenter un de ces agents.

Représentant « Balade » (j)

Le concept de « balade » nous semble approprié pour désigner une activité qui consiste à se déplacer tout en évitant les obstacles. Pour se balader il faut donc pouvoir se déplacer, et donc faire appel au représentant « motricité » (l'agent g), mais surtout éviter de rencontrer des obstacles qui ralentiraient ou empêcheraient le déplacement.

C'est pourquoi le représentant « balade » (l'agent j sur la figure 4.28), a besoin de percevoir les obstacles et fait donc appel pour cela au représentant « obstacle », l'agent f présenté précédemment. Nous voyons donc ce représentant comme un filtre qui proposera les mêmes actions que le représentant moteur, mais qui en fonction de la perception d'obstacle ou pas, choisira le déplacement le mieux approprié pour éviter

de rencontrer un obstacle. C'est ce comportement qu'illustrent les captures d'écran présentées en figure 4.32.

Au début (figure 4.32.a), le représentant balade a reçu l'ordre d'avancer, ordre qu'il répercute sur le représentant motricité qu'est l'agent g. Puis le robot arrive sur un obstacle, figure 4.32.b. L'agent j reçoit toujours l'ordre d'avancer, mais le représentant obstacle (l'agent f) renvoie la perception $^7/_8$ correspondant à la perception « obstacle devant gauche ». Le représentant balade choisit donc d'envoyer l'ordre « tourner à droite » au représentant motricité comme le montre la figure 4.32.c. Lorsque il n'y a plus d'obstacle perçu, le représentant balade se remet à relayer l'ordre « avancer » qu'il reçoit, et le robot avance tout droit (figure 4.32.d).

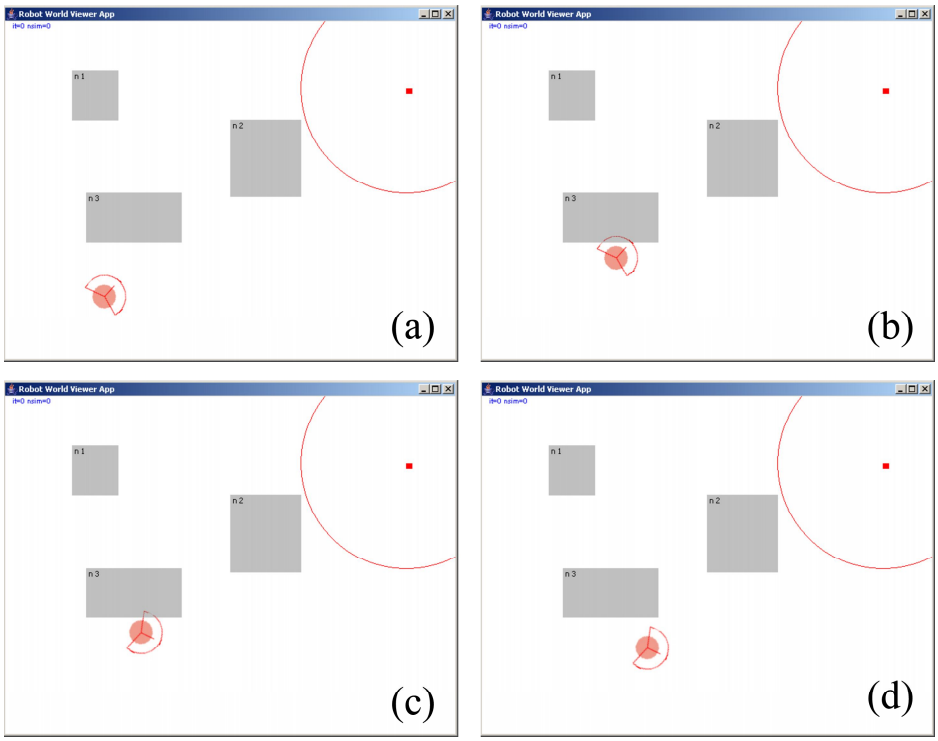

FIG. 4.32 – Figures montrant le comportement du représentant balade qui a reçu l'ordre d'avancer (a). Lorsqu'il perçoit un obstacle (b), il prend l'initiative de tourner plutôt que de continuer à avancer (c). Quand il ne perçoit plus d'obstacle, il se remet à demander au représentant motricité d'avancer (d).

A présent, observons la matrice d'action de cet agent représentant qui est donnée par la table 4.7. Nous pouvons voir que l'agent j propose les même cinq ordres que l'agent g. Lorsqu'aucun obstacle n'est perçu, c'est-à-dire lorsque $g = ^1/_8$, l'agent j répercute l'ordre reçu sur l'agent g (ce sont les lignes 1, 5 et 9 de la table 4.7). Par contre, lorsqu'un obstacle est détecté dans la direction de l'ordre de déplacement, l'ordre envoyé au représentant « motricité » tient compte de la perception d'un obstacle.

Penchons-nous sur la ligne 8 de la table 4.7 : quand l'agent j reçoit l'ordre de tourner à gauche ($j = 3/8$), mais qu'il perçoit un obstacle devant lui à gauche ($g = 7/10$), il préférera donner l'ordre de tourner à droite au représentant « motricité » ($j = 3/8$). Enfin, l'ordre « reculer » (la dernière ligne de la table) est appliquée quelle que soit la perception reçue. En effet nous n'avons intégré aucun capteur qui ne permette de percevoir les obstacles se trouvant derrière le robot.

TAB. 4.7 – Matrice d'action de l'agent j (Représentant « Balade »).

DA	Perception	Action	Légende
1/8	(f=1/8)	(g=5/10)	Avancer
1/8	(f=3/8)	(g=3/10)	Avancer en tournant à gauche
1/8	(f=5/8)	(g=1/10)	« Avancer » en reculant
1/8	(f=7/8)	(g=7/10)	Avancer en tournant à droite
3/8	(f=1/8)	(g=3/10)	Tourner à gauche
3/8	(f=3/8)	(g=3/10)	Tourner à gauche
3/8	(f=5/8)	(g=3/10)	Tourner à gauche
3/8	(f=7/8)	(g=7/10)	« Tourner à gauche » en tournant à droite
5/8	(f=1/8)	(g=7/10)	Tourner à droite
5/8	(f=3/8)	(g=3/10)	« Tourner à droite » en tournant à gauche
5/8	(f=5/8)	(g=7/10)	Tourner à droite
5/8	(f=7/8)	(g=7/10)	Tourner à droite
5/8	∀ f	(g=9/10)	Reculer

Représentant « Relation à la base » (i)

Les agents présentés jusqu'à présent permettent au robot de se déplacer sans rencontrer d'obstacles. Nous pouvons donc utiliser les représentants ainsi créés pour déplacer le robot vers un endroit qui présente un intérêt pour lui : à savoir la base lui permettant de se recharger. Pour accomplir cette tâche qui consiste à s'approcher de la base, ou à s'en éloigner, nous créons le représentant « relation à la base », l'agent i sur la figure 4.28.

Cet agent va donc devoir se diriger dans la direction de la base en donnant des ordres à l'agent j responsable de la « balade » en fonction de la direction dans laquelle est perçue la base. Pour cela l'agent i a donc besoin des perceptions de l'agent h qui est l'agent capteur de la direction de la base.

Le comportement attendu est illustré par les captures d'écran présentées en figure 4.33. Lorsque l'expérience commence (figure 4.33.a), le représentant « relation à la base » (l'agent i) reçoit l'ordre d'approcher de la base. Comme l'agent h l'informe qu'il perçoit

la base devant lui, l'agent i demande au représentant balade (l'agent j) d'avancer. L'agent j continue de transmettre l'ordre d'avancer au représentant moteur, jusqu'à ce qu'un obstacle soit perçu (figure 4.33.b). Dès lors le représentant balade fait tourner le robot à droite pour lui faire éviter l'obstacle (figure 4.33.c). A force de tourner, l'agent h envoie la valeur $^3/_{10}$ qui correspond à la perception « base à gauche », figure 4.33.d. Le représentant « relation à la base », demande donc à l'agent j de tourner à gauche. Or, l'agent j reçoit de l'agent f la perception « obstacle à gauche », c'est pourquoi il transforme l'ordre « tourner à gauche » en « tourner à droite ». Enfin, lorsque l'agent f ne perçoit plus d'obstacle, l'agent j se remet à transmettre l'ordre de tourner à gauche, et le robot s'oriente à nouveau dans la direction de la base (figure 4.33.e), puis reprend sa tâche d'approche en avançant (figure 4.33.f).

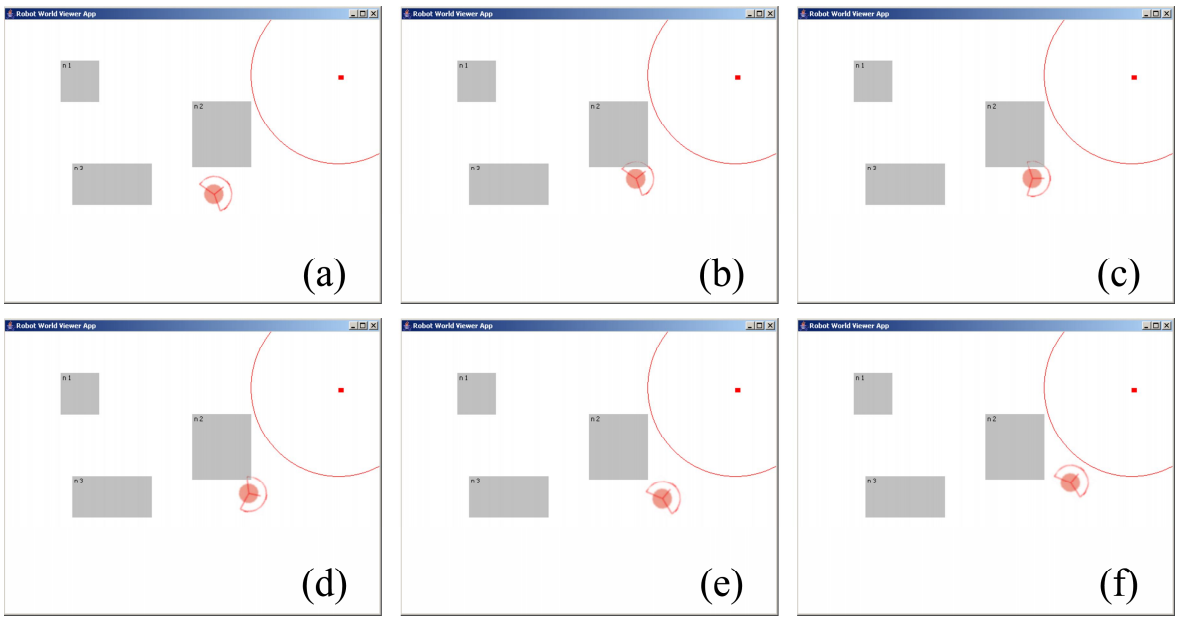

FIG. 4.33 – Captures d'écran montrant le comportement du robot lorsque l'on demande au représentant i de s'approcher de la base. Lorsqu'un obstacle est perçu, figures (b) et (c), l'agent j s'occupe de faire éviter l'obstacle. L'obstacle évité, le robot se réoriente vers la base (figure (e)) et reprends son approche (figure (f)).

Si nous observons la matrice d'action de cet agent représentant i (donnée par la table 4.8), nous remarquons que pour l'ordre « s'éloigner » ($i = ^2/_3$), et pour la perception « base devant » ($h = ^5/_{10}$) l'agent i choisira aléatoirement l'action « tourner à gauche » ($j = ^3/_8$) ou l'action « tourner à droite » ($j = ^5/_8$). En effet, comme nous n'avons pas intégré de capacité à percevoir les obstacles derrière le robot, il est peu judicieux de vouloir le faire reculer, et il est donc préférable de le faire se mettre dos à la base pour pouvoir plus tard s'en éloigner en marche avant. Pour ce faire, le sens de rotation du robot n'a aucune importance, c'est pourquoi, pour un même ordre et une même

perception, les actions « tourner à gauche » et « tourner à droite » auront la même probabilité d'être déclenchées.

TAB. 4.8 – Matrice d'action de l'agent i (Représentant « Relation à la base »).

DA	Perception	Action	Légende
$1/3$	(h=$1/10$)	(j=$3/8$)	Approcher en tournant à gauche
$1/3$	(h=$3/10$)	(j=$3/8$)	Approcher en tournant à gauche
$1/3$	(h=$5/10$)	(j=$1/8$)	Approcher en avançant
$1/3$	(h=$7/10$)	(j=$5/8$)	Approcher en tournant à droite
$1/3$	(h=$9/10$)	(j=$5/8$)	Approcher en tournant à droite
$2/3$	(h=$1/10$)	(j=$1/8$)	S'éloigner en avançant
$2/3$	(h=$3/10$)	(j=$5/8$)	S'éloigner en tournant à droite
$2/3$	(h=$5/10$)	(j=$3/8$) ou (j=$5/8$)	S'éloigner en tournant à gauche ou à droite
$2/3$	(h=$7/10$)	(j=$3/8$)	S'éloigner en tournant à gauche
$2/3$	(h=$9/10$)	(j=$1/8$)	S'éloigner en avançant

Délégué à l'émotion « Jeux » (l)

Jusqu'à présent, nous avons présenté la couche physique, puis les représentants qui s'appuient sur celle-ci pour proposer des tâches de haut niveau que l'on peut considérer comme des procédures. Nous allons maintenant présenter les agents de la couche émotionnelle qui vont disposer des agents représentants pour satisfaire les émotions ressenties par le système.

Commençons par l'agent l délégué à l'émotion de « jeu » et dont la matrice d'action est présentée dans la table 4.9. Nous rappelons que l'émotion de jeu est satisfaite lorsque le robot se déplace, c'est pourquoi lorsque l'émotion de « jeu » est peu satisfaite ($l = 3/6$) ou très insatisfaite ($l = 1/6$), les actions choisies seront celles qui offriront un déplacement maximal, à savoir « avancer » ($j = 1/8$) ou « reculer » ($j = 7/8$). Par contre lorsque l'émotion de « jeu » est satisfaite, le robot n'ayant pas besoin de se déplacer rapidement, l'agent l pourra choisir de tourner d'un coté ou de l'autre.

TAB. 4.9 – Matrice d'action de l'agent l (Délégué au jeu).

DA	Perception	Action	Légende
$1/6$		(j=$1/8$) ou (j=$7/8$)	Avancer ou reculer
$3/6$		(j=$1/8$) ou (j=$7/8$)	Avancer ou reculer
$5/6$		(j=$3/8$) ou (j=$5/8$)	Tourner à gauche ou à droite

Délégué à l'émotion « Faim » (k)

Pour le moment, notre robot est donc capable de se déplacer dans l'environnement en satisfaisant son émotion de « jeu ». Il est donc temps de lui permettre de satisfaire une émotion bien plus importante à savoir la « faim ». En effet cette émotion est directement liée à l'état de charge des batteries et des batteries vides auraient pour impact l'arrêt du système.

C'est pourquoi nous présentons ici le délégué à l'émotion de « faim ». Ce dernier a pour rôle de faire approcher le robot de la base lorsque l'état de charge des batteries est jugé préoccupant, comme le montre la matrice d'action donnée par la table 4.10.

TAB. 4.10 – Matrice d'action de l'agent k (Délégué à la faim).

DA	Perception	Action	Légende
$^1/_6$		(i=$^1/_3$)	Approcher
$^3/_6$		(i=$^1/_3$)	Approcher
$^5/_6$		(i=$^2/_3$)	S'éloigner

Gestionnaire d'émotions (m)

Nous arrivons à présent à l'élément le plus intéressant du système : le gestionnaire d'émotion, représenté par l'agent m sur la figure 4.28. En effet cet agent va devoir choisir quelle émotion satisfaire afin de conserver un état émotionnel convenable. Pour ce faire, le gestionnaire d'émotions va donc choisir à quel délégué il va faire appel, et donc quelle émotion il choisit de satisfaire. Dans le cas présent, le gestionnaire d'émotions subit deux émotions : le « jeu » et la « faim », de ce fait il dispose de deux délégués qui sont, respectivement, les agents l et k.

Comme pour tous les agents présentés jusqu'à présent, la politique d'action du gestionnaire d'émotions est donnée par sa matrice d'action. Nous proposons pour cet agent deux matrices d'action présentées en table 4.11 et 4.12. Comme on peut le constater, ces deux matrices d'action diffèrent par leur contenu et cela a un impact sur le comportement du système.

Si l'on observe la première matrice (table 4.11), on remarque que la priorité est donnée à la satisfaction de l'émotion de « faim ». C'est-à-dire que, dès que l'émotion de fin descendra en dessous d'une valeur de $^2/_3$, le gestionnaire d'émotions engagera une procédure visant à recharger les batteries en faisant appel au délégué à l'émotion de « faim », l'agent k. C'est pourquoi nous avons employé le terme « peureux » : en effet, le comportement attendu est que le robot cherche à se recharger souvent. De cette

manière, on donne au robot le maximum de chances d'éviter une décharge complète des accumulateurs.

TAB. 4.11 – Matrice d'action de l'agent m (Gestionnaire d'émotions) : version « peureux ».

DA	Perception	Action	Légende
	(k=1/6, l= 1/6)	(k=1/6)	Satisfaire la faim
	(k=1/6, l= 3/6)	(k=1/6)	Satisfaire la faim
	(k=1/6, l= 5/6)	(k=1/6)	Satisfaire la faim
	(k=3/6, l= 1/6)	(k=1/6)	Satisfaire la faim
	(k=3/6, l= 3/6)	(k=1/6)	Satisfaire la faim
	(k=3/6, l= 5/6)	(k=1/6)	Satisfaire la faim
	(k=5/6, l= 1/6)	(l=1/6)	Satisfaire le jeu
	(k=5/6, l= 3/6)	(l=1/6)	Satisfaire le jeu
	(k=5/6, l= 5/6)	(l=1/6)	Satisfaire le jeu

A présent, intéressons-nous à la seconde matrice donnée par la table 4.12. Cette fois-ci, on peut remarquer que la priorité est donnée à la satisfaction de l'émotion de jeu. En effet, le gestionnaire d'émotions ne déclenchera la procédure « aller se recharger » que lorsque les batteries auront un état de charge critique (dans le cas présent, un niveau de charge inférieur à 1/3). Nous avons donc choisi le terme de « joueur » pour qualifier le comportement engendré par cette politique car le robot donnera la priorité au jeu et attendra plus longtemps avant d'entamer la procédure de recharge.

TAB. 4.12 – Matrice d'action de l'agent m (Gestionnaire d'émotions) : version « joueur ».

DA	Perception	Action	Légende
	(k=1/6, l= 1/6)	(k=1/6)	Satisfaire la faim
	(k=1/6, l= 3/6)	(k=1/6)	Satisfaire la faim
	(k=1/6, l= 5/6)	(k=1/6)	Satisfaire la faim
	(k=3/6, l= 1/6)	(k=1/6)	Satisfaire le jeu
	(k=3/6, l= 3/6)	(k=1/6)	Satisfaire le jeu
	(k=3/6, l= 5/6)	(k=1/6)	Satisfaire le jeu
	(k=5/6, l= 1/6)	(l=1/6)	Satisfaire le jeu
	(k=5/6, l= 3/6)	(l=1/6)	Satisfaire le jeu
	(k=5/6, l= 5/6)	(l=1/6)	Satisfaire le jeu

Laquelle de ces politiques est la meilleure ? En fait, tout dépend de la capacité totale de charge des accumulateurs et de la taille de l'environnement. En effet, plus l'environnement est grand et parsemé d'obstacles, plus il est important d'entreprendre

la procédure de recharge tôt. Dans le cas contraire, c'est-à-dire si les batteries sont de taille suffisante par rapport à la taille de l'environnement, la seconde politique sera plus appropriée, car le robot interrompra moins souvent son activité de jeu pour entreprendre la tâche visant à aller se recharger.

Nous ne pouvons donc pas, avant d'avoir plongé le robot dans son environnement, choisir une politique plutôt qu'une autre. Le risque à cela est de se retrouver avec un robot inadapté à son environnement.

Dans la section qui suit, nous allons donc voir comment une telle structure de contrôle va pouvoir émerger à partir des couches physique et émotionnelle, et comment cette structure va évoluer afin de s'adapter à l'environnement dans lequel elle est plongée.

4.2.7 Discussion

Nous venons de présenter un modèle d'architecture de contrôle où la structure de contrôle est constituée d'agents. Nous avons également vu comment les valeurs contenues dans les matrices d'action permettaient d'exprimer un comportement tel que l'évitement d'obstacle ou le déplacement vers la base de recharge. Quant à l'utilisation d'émotions, nous avons vu que cela permettait au robot de basculer et donc de satisfaire deux besoins tels que jouer et recharger ses accumulateurs.

Outre le fait que l'architecture présentée en sous-section 4.2.6 réalise bien ce pour quoi nous l'avons créé, nous pouvons remarquer que les différents agents utilisés sont comparables à des procédures que l'on peut assembler pour la réalisation d'une architecture de contrôle. Ainsi on peut voir notre modélisation comme un *langage de programmation* permettant de définir d'une manière assez intuitive la structure de contrôle d'un robot. En effet l'organisation des agents représentants au sein de la structure de contrôle peut être vue comme un assemblage de concepts permettant de réaliser des abstraction des capteurs et des effecteurs de la couche physique. Le concepteur pourra donc, comme avec un langage de programmation conventionnel, commencer par définir des concepts simples issus des capteurs et effecteurs. Puis il pourra construire, grâce à ces premières abstractions de la couche physique, des concepts de plus haut niveau et des tâches de plus en plus complexes jusqu'à l'obtention d'une structure de contrôle complète.

Cependant, les outils que nous avons présenté en début de ce chapitre, comme les émotions et l'apprentissage par renforcement, n'ont pas été exploités à fond. En effet, nous n'avons utilisé de l'apprentissage par renforcement que les structures contenant les poids (les matrices d'action) et permettant donc de réaliser un processus de sélection d'action. Ce serait oublier qu'initialement nous avions conçu ces matrices d'action dans

la perspective de pour pouvoir faire évoluer ces poids de déclenchement. Concernant les émotions, elle n'ont été jusqu'ici utilisées que pour définir une politique de sélection d'action alors que celles-ci pourraient être utilisées pour guider la manière dont se construit la structure de contrôle.

Nous allons donc proposer dans le chapitre suivant des méthodes permettant de faire émerger une structure de contrôle adaptée aux besoins du robot, c'est-à-dire permettant de satisfaire les émotions auxquelles le robot est assujetti.

Chapitre 5

Construction de la structure de contrôle et apprentissage

Comme nous l'avons vu précédemment, la structure de contrôle du robot doit être adaptée à la structure physique du robot, et doit évoluer pour apprendre et réaliser des tâches de plus en plus complexes. C'est pourquoi l'apprentissage à l'intérieur du système multi-agents est réalisé en deux phases : la première, le babbling learning, consiste à regrouper les capteurs et les effecteurs similaires (dans le sens où ils sont physiquement proches, ou s'ils ont le même mode de fonctionnement). La deuxième phase, le satisfying learning (apprendre à se satisfaire), utilise des groupements déjà créés, notamment pendant la phase de babbling, comme base pour construire des connaissances plus abstraites.

5.1 La première phase : le babbling learning

5.1.1 Similarité d'impact sur les capteurs de deux moteurs

Lors de la phase de babbling, les effecteurs ayant le même impact sur les capteurs sont regroupés entre eux. Ce regroupement prend également en compte l'impact émotionnel des actions sur l'agent.

Dans cette sous-section nous allons rapidement présenter comment nous calculons l'*impact d'un moteur* sur les capteurs d'un robot. Pour cela nous modélisons l'impact d'un moteur sur n capteur sous la forme d'un vecteur à n dimensions. Puis, nous nous appuyons sur les calculs de distance et d'angle pour définir le coefficient de similarité qui nous permet de grouper deux moteurs ayant un fonctionnement similaire.

Impact d'un moteur sur les capteurs

L'impact d'un moteur m_k sur les n capteurs c_j est un vecteur de dimension n où chaque composante j représente la variation de la valeur v du capteur c_j pendant que l'on stimule le moteur m_i. Ainsi, la variation d'un capteur c_j à l'instant t est donnée par :

$$\Delta_t^{c_j} = \sum_{i=0}^{t} |v_i - v_{i-1}| * |\text{GSat}_i - \text{GSat}_{i-1}| \tag{5.1}$$

On peut noter que dans cette équation intervient le calcul de la variation de l'état émotionnel du robot, noté GSat. Cela permet de donner plus d'importance aux variations associées à des changements importants de l'état émotionnel du robot.

A partir de là, on définit donc le vecteur $\overrightarrow{\mathrm{I}_t^{m_k}}$ qui représente l'impact d'un moteur m_k sur l'ensemble des n capteurs à l'instant t :

$$\overrightarrow{\mathrm{I}_t^{m_k}} = \begin{pmatrix} \Delta_t^{c_1} \\ \Delta_t^{c_2} \\ \vdots \\ \Delta_t^{c_n} \end{pmatrix} \tag{5.2}$$

Coefficient de similarité

Pour déterminer si l'impact $\vec{V_k}$ du moteur m_k est proche de l'impact $\vec{V_1}$ du premier moteur testé m_1, il faut que :

- la distance entre $\vec{V_1}$ et $\vec{V_k}$ soit faible,
- l'angle formé par les deux vecteurs soit faible.

C'est pourquoi le calcul du coefficient de similarité σ prend en compte le coefficient de similarité des distances σ_d et d'angle σ_θ :

$$\sigma = \sigma_d \cdot \sigma_\theta \tag{5.3}$$

Le coefficient de similarité des distances est le nombre $\sigma_d \in [0, 1]$ donné par le ratio entre la plus petite norme et la plus grande :

$$\sigma_d = \frac{min(\left\|\vec{\mathrm{V}_1}\right\|, \left\|\vec{\mathrm{V}_2}\right\|)}{max(\left\|\vec{\mathrm{V}_1}\right\|, \left\|\vec{\mathrm{V}_2}\right\|)} \tag{5.4}$$

Pour le calcul du coefficient de similarité d'angle $\sigma_\theta \in [0, 1]$, il faut prendre en considération que si l'angle formé par les deux vecteurs est proche de π, cela signifie

que les deux vecteurs sont radicalement opposés. Par contre lorsque l'angle vaut $2 \cdot \pi$, les vecteurs sont parallèles. Le coefficient de similarité d'angle est donc inversement proportionnel à la distance de l'angle θ à π. Ramené à l'intervalle $[0, 1]$, cela donne :

$$\sigma_\theta = \begin{cases} 1 - \frac{\theta}{\pi} & \forall \theta \leq \pi \\ 1 - \frac{2 \cdot \pi - \theta}{\pi} & \forall \theta > \pi \end{cases} \tag{5.5}$$

Les calculs de ces coefficients permettent donc de déterminer si deux moteurs ont des effets similaires sur les perceptions du robots et s'ils doivent donc êtres regroupés et présentés à la structure de controle à travers un agent *représentant*. Ces calculs sont présentés plus en détails en annexe A.

5.1.2 Fonctionnement du babbling learning

Le babillage est l'acte qui consiste, pour le bébé, à produire des sons, d'abord de manière accidentelle, puis à reproduire certains d'eux de manière régulière et répétée. A la fin de la première année, le babillage devient plus clair et quelques sonorités sont répétées intentionnellement, l'enfant a alors les capacités à prononcer ses premiers mots.

La phase de babbling learning consiste donc à générer des mouvements aléatoires et à observer leurs impacts sur les perceptions, comme mentionné dans [Frolov *et al.* , 2002]. Pour réaliser ceci, le gestionnaire d'émotions envoie des valeurs aléatoires aux effecteurs, et observe leurs impacts sur l'environnement et la structure physique du robot en écoutant les messages envoyés par les agents capteurs et en utilisant les notions de similarité d'impact présentées dans la sous-section précédente.

On associe, par ce processus, le vecteur cible (le vecteur des mouvements perçus en entrée) au vecteur des commandes motrices générées. Ainsi les moteurs ayant des impacts similaires sur les capteurs seront regroupés entre eux.

A la fin de cette phase de babillage, on obtient une structure de contrôle qui est proche du schéma moteur du robot et qui prend en compte les contraintes physiques et la dynamique du système, comme sur la Figure 5.1.

Sur la gauche de la figure, tous les capteurs et les moteurs (i.e. les roues droite et gauche) appartiennent au même groupe, la couche physique. Le gestionnaire de groupe (l'agent en blanc sur la figure, qui n'est d'autre que le gestionnaire d'émotions) stimule les effecteurs (i.e. les moteurs des roues droite et gauche), et observe les nouvelles valeurs envoyées par les capteurs. Quand le gestionnaire remarque une similarité de fonctionnement, il crée un nouveau groupe correspondant.

Sur la partie droite de la Figure 5.1, les effecteurs similaires ont été regroupés dans ce que l'on peut considérer comme le groupe responsable du déplacement. Un agent représentant est alors placé dans le groupe dans le but d'interagir avec les couches plus

hautes en utilisant un nombre limité d'ordres synthétisés (par exemple avancer, reculer, tourner à droite, à gauche).

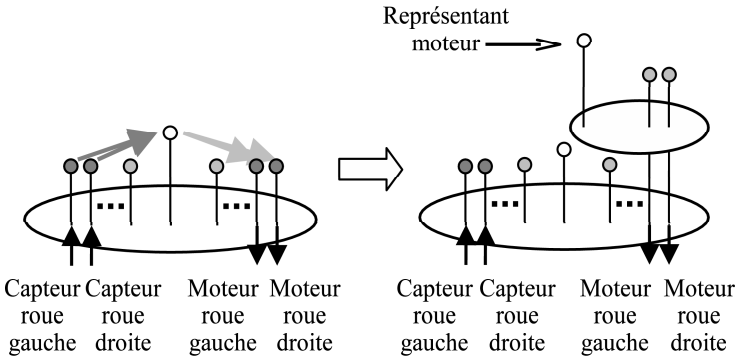

FIG. 5.1 – La phase de babbling. A gauche, le gestionnaire d'émotions stimule les moteurs et observe la réaction des capteurs. A droite, lorsqu'une similarité de fonctionnement est découverte, un nouvel agent est créé pour représenter et regrouper les deux moteurs concernés.

Lors de la création de ce représentant celui-ci va obtenir deux objets pour défuzzyfier ses actions. Ces deux objets, encerclés en pointillés sur la partie gauche de la figure 5.2, vont lui permettre de discerner plusieurs actions possibles. Ainsi, lorsque le représentant choisira une action celle-ci sera convertie en grandeurs réelles à appliquer sur les agents moteur.

Un représentant créé lors de la phase de babbling va donc proposer une combinaison des actions possibles sur chaque agent qu'il représente. D'autres agents pourront alors lui demander d'exécuter une de ces actions en lui envoyant une grandeur réelle qui sera fuzzyfiée pour être convertie en un état. Toutes les actions dont nous venons de parler sont listées sur la partie droite de la figure 5.2 qui présente les trois mêmes agents. Ce nombre d'action est donné par un découpage, arbitraire, en trois sous-ensembles flous de l'espace de variation de chacun des moteurs. Le représentant moteur va donc proposer $3^2 = 9$ actions possibles.

Enfin, ce représentant est soumis à une émotion : celle de l'agent qui l'a créé. Or c'est le gestionnaire d'émotions qui en exécutant la phase de babbling est amené à créer ce représentant. C'est donc l'émotion à laquelle est soumis le gestionnaire d'émotion, à savoir $GSat$, qui est envoyée à ce représentant.

Dans nos travaux nous nous sommes limités, pour la phase de babbling, à la création de représentants moteurs. Mais pour réaliser une structure de contrôle qui soit plus proche du schéma sensori-moteur il serait intéressant de regrouper également les capteurs au fonctionnement similaire. C'est ce que présente la figure 5.3 sur laquelle on peut voir un agent qui représente les capteurs ainsi qu'un agent qui permet de mettre

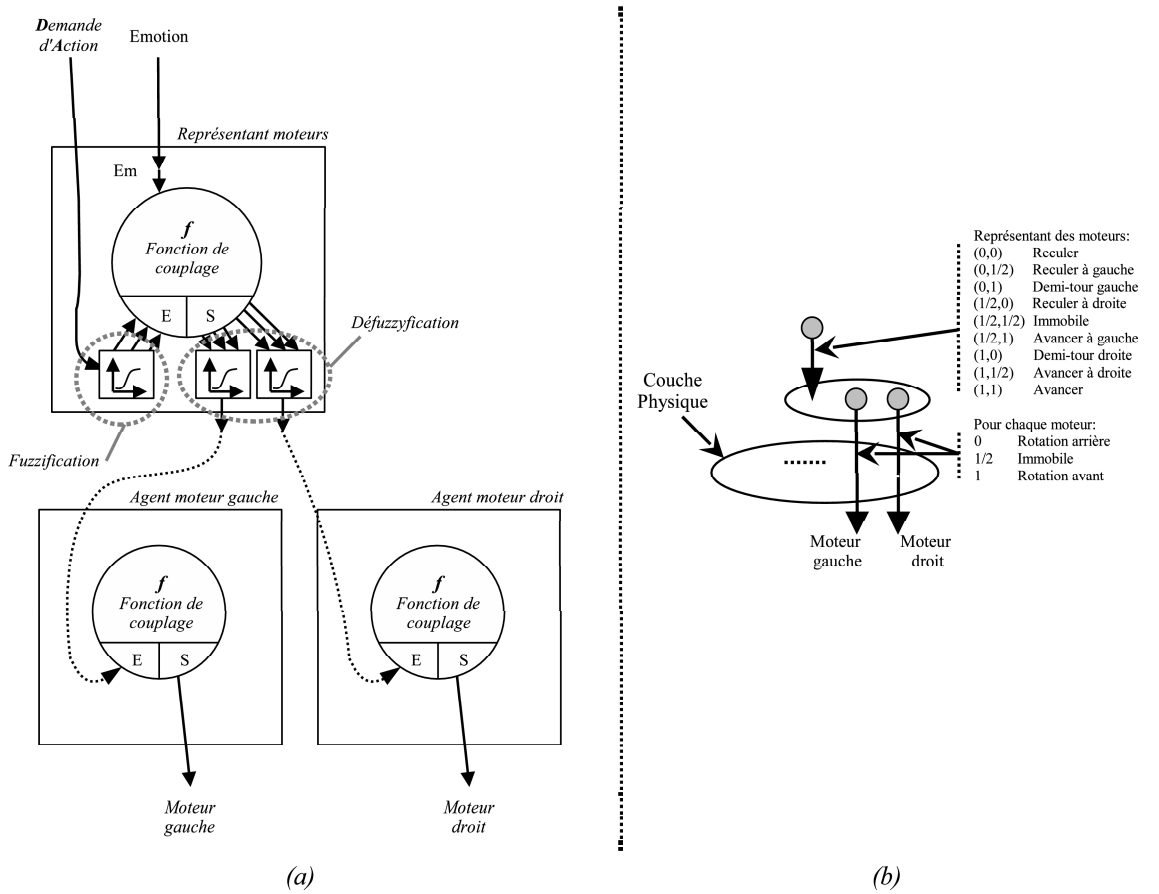

FIG. 5.2 – Illustrations de l'agent représentant créé durant la phase de babbling, et des deux agents moteurs qu'il représente. A gauche (a), on peut voir les strucutures des trois agents et leurs connections entre eux. A droite (b), l'organisation correspondante et les différentes actions qui pourront êtres demandées à chaque agent.

en relation le représentant capteur avec le représentant moteur. Ce dernier pourrait permettre de piloter les roues en garantissant le déplacement qui a été demandé. En pratique, c'est le *satisfying learning*, présenté dans la section 5.2 qui suit, qui va créer un représentant capteur, puis un agent qui coordonnera le représentant capteur avec le représentant moteur.

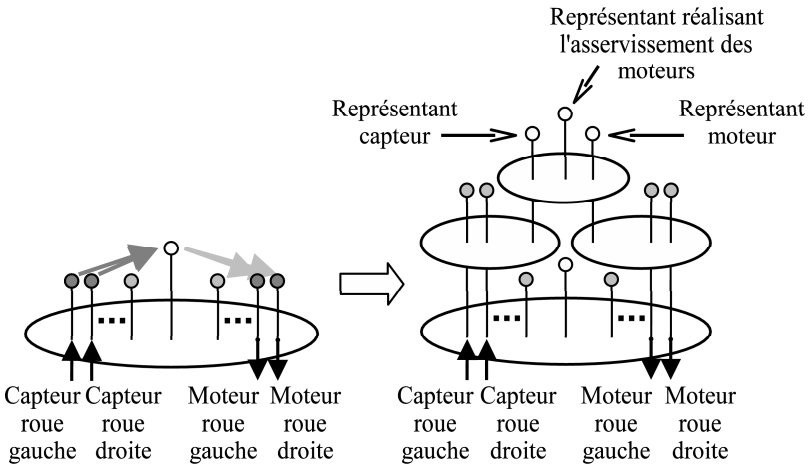

FIG. 5.3 – Un babbling plus performant consiterait à regrouper également entre eux les capteurs ayant un fonctionnement similaire. Lorsqu'une similarité de fonctionnement est découverte, il faudrait créer également un agent représentant les capteurs concernés. Enfin, un troisième agent pourrait alors être créé afin de représenter les deux premiers représentants et permettant de réaliser un asservissement des moteurs.

5.2 La deuxième phase : le satisfying learning

Qu'est ce que « se satisfaire » ?

Nous avons présenté précédemment ce qu'était une émotion, et nous avons alors défini ce que représentaient la satisfaction et l'insatisfaction d'une émotion. « Se satisfaire » consiste donc à faire augmenter le niveau de satisfaction d'une émotion ou à éviter que celui-ci ne baisse. Dans les deux cas, soit l'évolution d'une émotion est la conséquence d'une action entreprise, soit la suite logique d'une perception.

Il faut donc être capable de détecter une situation insatisfaisante avant qu'elle ne se produise et, si elle se produit, être capable de mettre fin à cette insatisfaction.

Apprendre à se satisfaire ... Comment faire ?

La tâche « se satisfaire » se réalise en accomplissant des actions satisfaisantes, et en étant capable d'anticiper, c'est-à-dire prévoir une insatisfaction et adopter le comportement adéquat. Dans cette phase, on utilise les abstractions des couches basses, notamment celles faites pendant la phase de babbling, pour construire des concepts de plus haut niveau. On va alors construire, par complexité croissante, tous les concepts permettant de réaliser une politique d'action qui amène à un état de satisfaction. Chaque nouveau concept appris peut être considéré par la suite comme une primitive qui peut être utilisée pour la construction d'autres concepts plus abstraits.

Pour réaliser cette phase d'apprentissage, il faut donc construire de nouveaux représentants qui vont venir s'intercaler entre la couche physique et la couche émotionnelle. En parallèle de cette activité de construction, tous les représentants impliqués dans la réalisation d'une tâche vont renforcer les poids de déclenchement de leurs matrices d'action en fonction de l'évolution de l'émotion à laquelle ils sont rattachés.

Nous allons donc montrer dans un premier temps, en sous-section 5.2.1, comment cette construction de la structure de contrôle est réalisée, et par quel agent. Puis, dans un second temps sera présenté en sous-section 5.2.4 le processus d'adaptation réalisé au sein des représentants au fur et à mesure que le robot agit et ressent des émotions.

5.2.1 Construction de la structure de contrôle

Le besoin de construire un représentant apparaît lors de l'évolution non prévue d'une émotion. Il nous a donc paru logique de laisser chaque délégué à une émotion trouver par lui-même les moyens de satisfaire son émotion. En effet, si on laissait cette tâche au gestionnaire d'émotions, celui-ci aurait à gérer en parallèle un apprentissage par émotion à satisfaire. De plus, comme nous allons le voir, la phase d'observation des perceptions et des actions qui amène à la construction d'un nouveau représentant n'a de sens que par rapport à une émotion. C'est pourquoi la construction de la structure de contrôle, et donc des représentants qui la constituent, est de la responsabilité de chaque délégué à une émotion.

Chacun des délégués cherche à satisfaire l'émotion à laquelle il est assigné. Il va donc chercher les situations qui lui permettent de se satisfaire. Quand le délégué remarque qu'un certain type de perception précède une chute de satisfaction, il mémorise cette configuration. Du point de vue du robot, une configuration capteur de ce type représente une chute d'émotion probable à laquelle il faut réagir pour satisfaire son émotion. De la même manière, une configuration moteur amenant du plaisir correspond à l'action à entreprendre pour obtenir satisfaction.

Pour mémoriser une configuration, le délégué crée un groupe avec les agents concernés et enregistre les valeurs de chacun d'entre eux. Un agent représentant est alors placé dans ce groupe pour produire le comportement souhaité ou pour donner la perception intéressante. Au fur et à mesure que le robot évolue et se déplace, il va apprendre les objectifs intermédiaires et les actions à entreprendre qui lui permettent de satisfaire son émotion.

Le rôle d'un délégué est donc de :

- choisir, pour une situation émotionnelle donnée, les comportements qui amènent systématiquement du plaisir : c'est-à-dire des actions qui amèneront directement à la satisfaction d'une émotion (voir figure 5.5). C'est le mode « Rechercher une satisfaction immédiate » illustré sur la figure 5.4, et dont le fonctionnement est présenté en sous-section 5.2.2.
- construire et intégrer dans la structure des représentants qui permettront par la suite au délégué d'anticiper pour obtenir du plaisir (voir figures 5.7 et 5.8). Cela correspond aux deux modes « Rechercher P_{post} et A_{curr} » et « Rechercher P_{pre} » présentés sur la figure 5.4 et détaillés dans la sous-section 5.2.3.

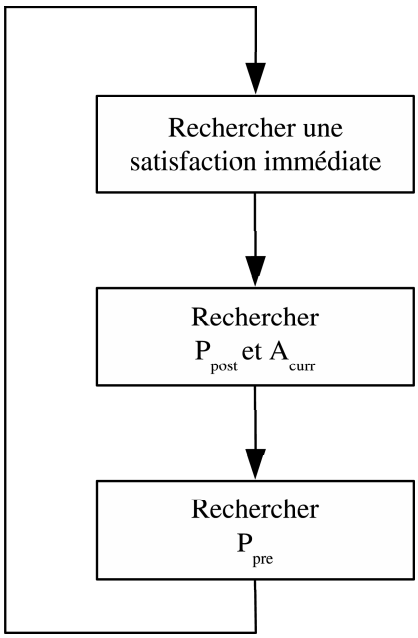

FIG. 5.4 – Modes de fonctionnement d'un délégué à une émotion. L'agent passe au mode suivant lorsque la recherche est fructueuse ou après un certain temps passé à persister dans ses recherches.

5.2.2 Réaction à une chute d'émotion

Pour choisir en fonction d'une situation émotionnelle donnée le comportement le plus approprié, le délégué doit avoir une matrice d'action qui, pour chaque couple (perception,action), définit un poids de déclenchement proportionnel aux récompenses obtenues précédemment. Nous rappelons que la récompense est donnée par rapport à l'évolution de l'émotion. Ainsi, une action qui fait décroître l'émotion a un poids de déclenchement plus faible qu'une action qui laisse l'émotion stable (c'est-à-dire qui n'affecte pas l'émotion). Et enfin, une action qui n'affecte pas l'émotion a un poids de déclenchement plus faible qu'une action qui augmente la valeur de l'émotion. Chaque choix d'action aura donc un impact sur le poids associé, ce qui permet d'apprendre à mieux sélectionner tout en continuant à agir.

Mais pour pouvoir disposer de perceptions à prendre en compte et d'actions à choisir, il faut bien sûr disposer de ces perceptions et de ces actions : c'est-à-dire construire des représentants qui vont générer les perceptions et proposer des actions. Cette phase qui consiste à créer un représentant qui proposera des actions satisfaisantes correspond au mode « Rechercher une satisfaction immédiate » illustré sur la figure 5.4. Le délégué ne peut être dans ce mode que lorsqu'il est activé par le gestionnaire d'émotions. En effet, si on laissait le délégué essayer des actions alors que le gestionnaire d'émotions active un autre délégué, deux représentants moteurs pourraient recevoir des ordres en simultané.

Recherche d'une action satisfaisante

Pour trouver une action qui amènera une satisfaction immédiate, le délégué va stimuler chacun des représentants moteurs qui existent, en commençant par ceux qui sont les plus éloignés de la couche physique. Il va donc stimuler à tour de rôle chaque représentant moteur. Ce processus est illustré sur la partie gauche de la figure 5.5.

Pour chaque Représentant Moteur RM_j, le délégué crée une matrice d'apprentissage qui, pour chaque action, permettra de connaître l'impact sur l'émotion à laquelle le délégué est rattaché. Cette matrice est constituée d'une seule colonne, et d'autant de lignes que le nombre d'action différentes que l'on peut demander à RM_j. Les poids w dans la matrice sont initialisés avec une valeur de 0,5, et le nombre d'essais pour chaque action fixé à 0.

Une fois la matrice créée, le délégué procède à une phase d'apprentissage qui consiste à essayer chacune des actions proposées par RM_j. Les poids contenus dans la matrice vont alors évoluer. Si une ou plusieurs actions essayées par le délégué satisfont l'émotion les poids correspondant à ces actions vont augmenter.

Le délégué arrête cette phase d'apprentissage lorsque :

– chaque action a été testé au moins 3 fois et toutes les actions ont un poids inférieur à 0.5. Cela signifie qu'aucune action proposée par RM_j ne satisfait l'émotion du délégué. Dans ce cas le délégué passe à un autre représentant, plus proche de la couche physique, et ce jusqu'à les avoir tous essayé.
– il existe au moins une action A_k telle que $w_k > 0,5$ et $t_k > 3$. Cela signifie qu'il existe au moins une action qui satisfait l'émotion, car les poids correspondant ont augmenté. Dans ce cas il faut créer un représentant qui aura pour rôle de renforcer les actions les plus satisfaisantes. Cette construction est détaillée dans la partie qui suit.

Construction d'un représentant

Lorsque le délégué à l'émotion a trouvé une action satisfaisante il crée un représentant RM_m pour lui permettre de se satisfaire (l'agent d sur la figure 5.5). La fonction de couplage f de ce représentant RM_m est la matrice d'apprentissage précédemment construite par le délégué et utilisée pour repérer RM_j (l'agent c sur la figure 5.5).

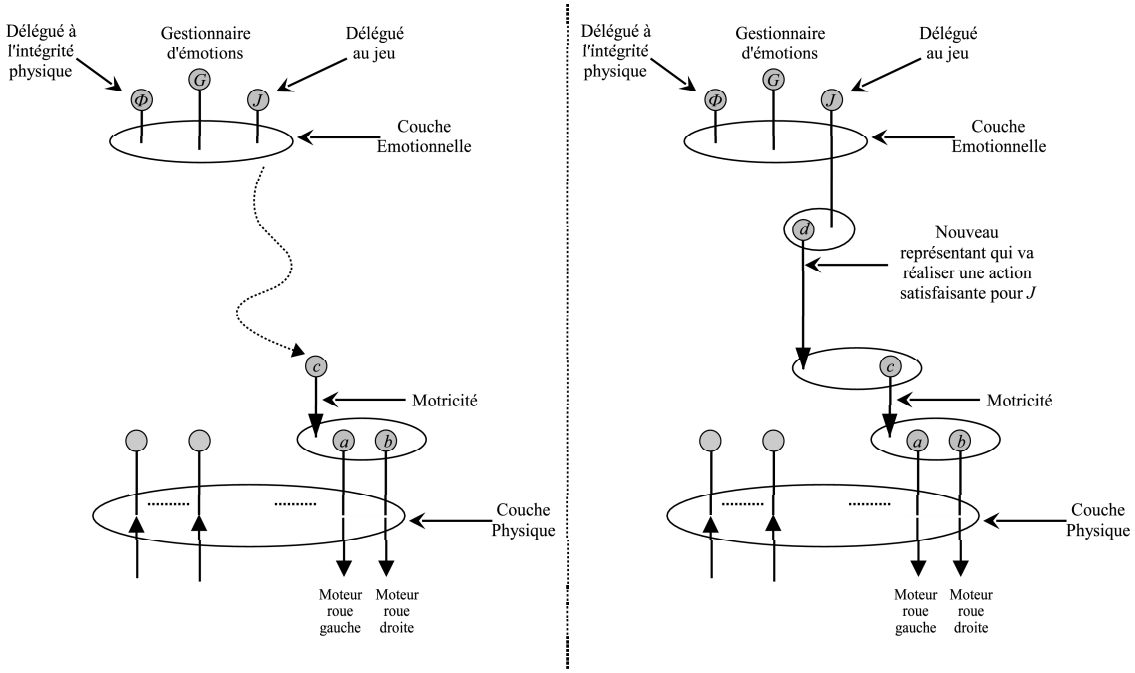

FIG. 5.5 – A gauche, le délégué au jeu, l'agent J a trouvé des actions satisfaisantes chez l'agent c. Avec ce qu'il a déjà appris, il crée à son image un représentant qui va être chargé de réaliser les actions satisfaisantes pour le délégué (à droite sur la figure).

Par la suite lorsque le délégué à l'émotion sera activé pour répondre à une insatisfaction, il fera appel à ce nouveau représentant qui, au fur et à mesure, renforcera les actions satisfaisantes. De ce fait les actions les plus satisfaisantes seront alors déclenchées plus souvent et le délégué à l'émotion trouvera dans le représentant qu'il a créé un

moyen efficace de satisfaire son émotion. Cela a pour effet de produire une structure de contrôle telle que celle présentée sur la partie droite de la figure 5.5 et qui sera en mesure de répondre de plus en plus efficacement aux insatisfations de l'émotion concernée.

5.2.3 Anticipation d'une chute d'émotion

Pour construire de nouveaux représentants et les intégrer dans la structure, il faut que la structure de contrôle découvre les actions qui font évoluer l'émotion. Le délégué doit donc pour cela analyser ce qu'il s'est passé *avant* que l'émotion n'évolue. Qu'est-ce que « ce qu'il s'est passé avant » ? Nous pensons que c'est un enchaînement de perceptions et d'actions qui amène systématiquement la même évolution d'émotion, et cet enchaînement est représenté par la figure 5.6.

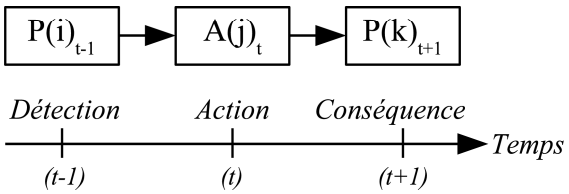

FIG. 5.6 – Enchaînement qui se produit avant l'évolution d'une émotion.

En effet, l'évolution d'une émotion se produit souvent en même temps qu'une perception donnée, notée $P(k)_{t+1}$. Si l'on remonte un peu dans le temps, on remarque que la perception $P(k)_{t+1}$ intervient après une action donnée, notée $A(j)_t$. Et en remontant encore dans le temps, on découvre qu'il existe une perception $P(i)_{t-1}$ qui intervient systématiquement avant l'action $A(j)_t$. Cette sorte de chaîne représente la suite logique qui amène à l'évolution de l'émotion. Prenons comme exemple le cas d'un chien qui aime jouer à la balle. Son comportement initial est : « si je vois une balle, je vais jouer avec car cela m'amène du plaisir ». Il vient à rencontrer un hérisson, qui pour lui ressemble à une balle. Lorsqu'il le touche pour la première fois, à la perception de la douleur (chute de l'émotion d'intégrité physique), il retire sa patte : cela produit une satisfaction immédiate, c'est ce que nous avons exposé dans la sous-section 5.2.2 qui précède.

Le chien commence donc à assimiler que la perception « patte sur des épines » est synonyme de douleur. Puis il cherche l'action en cause qui amène à l'état « patte sur le hérisson », et qu'il va donc falloir inhiber. Enfin, il découvre la perception qui précède cette action : à savoir « une balle avec des piquants ». Dès lors il sait que s'il perçoit une balle avec des piquants, puis qu'il essaye de jouer avec comme avec une balle normale, il va obtenir la perception « patte sur des épines » synonyme de douleur.

Il faut donc trouver, lorsque l'on est dans une phase de jeu et que l'on perçoit « une balle avec des piquants », une action différente de « jouer avec ». Ce processus qui consiste à repérer la perception qui précède ainsi que l'action en cause dans la chute d'émotion correspond au deuxième mode de fonctionnement d'un délégué et qui est représenté sur la figure 5.4. Puis il faut rechercher la perception qui précède cette chute d'émotion afin de pouvoir anticiper, cela correspond au troisième mode de fonctionnement présenté sur la figure 5.4. Nous allons donc présenter ces deux derniers modes en détaillant quelles méthodes et quelles structures nous utilisons.

Recherche de P_{post} et de A_{curr}

Dans cette partie nous allons montrer :

- comment on va déterminer la perception, notée P_{post}, associée à la chute d'émotion et qui donnera lieu à la création d'un représentant destiné à détecter cette perception, noté R_{post}.
- comment on va trouver le représentant moteur R_{curr} responsable de l'action, notée A_{curr}, associée à la chute d'émotion.

Pour commencer, rappelons que le délégué à une émotion peut fonctionner dans ce mode sans que le gestionnaire d'émotions ne l'ai activé. En effet, comme nous allons le voir, le délégué a besoin que d'autres agents stimulent les actionneurs et fassent changer l'état des capteurs pour déterminer la perception et l'action associées à la chute d'émotion.

Pour déterminer P_{post} et A_{curr} le délégué a besoin de determiner, respectivement, quels représentants capteur et quels représentants moteur sont impliqués lors de la chute de l'émotion. Pour cela il faut pouvoir se rappeler de l'état de ces représentant à chaque fois ou l'émotion chute. A cet effet, lorsque le délégué rentre dans ce mode de fonctionnement, il va créer deux tableaux VP_{post} et VA_{curr} qui contiendront, repectivement, pour chaque représentant capteur et pour chaque représentant moteur le couple (valeur moyenne, écart type).

Ainsi, lorsque le délégué est dans ce mode de fonctionnement et qu'intervient la chute de l'émotion, il met à jour les deux tableaux VP_{post} et VA_{curr}. Pour que l'on puisse tirer une conclusion de ces écoutes, il faut que l'on ai fait un nombre suffisant de mise à jour des deux tableaux. Dans l'algorithme présenté en fin de cette section, nous avons fixé le nombre d'écoutes à 100.

Lorsque l'on a suffisament accumulé d'écoutes on passe alors à l'analyse des deux tableaux VP_{post} et VA_{curr}. A cet instant, pour les représentants qui reçoivent ou envoient toujours la même valeur, la variable écart type contenue dans le tableau correspondant est égale à 0. Cela permet donc de connaitre les représentants systématiquement impliqués dans la chute de l'émotion et la valeur qu'ils avaient à cet instant.

Pour ce qui est de déterminer P_{post}, si dans VP_{post} il existe au moins un écart type égal à 0, le délégué analyse le tableau et commence par sélectionner tous les représentants capteurs pour lesquels l'écart type est de 0.

Parmi les représentants sélectionnés, le délégué supprime ceux qui sont représentés par un agent pour lequel l'écart type est également de 0. Cela permet de considérer seulement les représentants qui proposent les perceptions les plus abstraites, au lieu de considérer également tous les agents que ces représentants représentent.

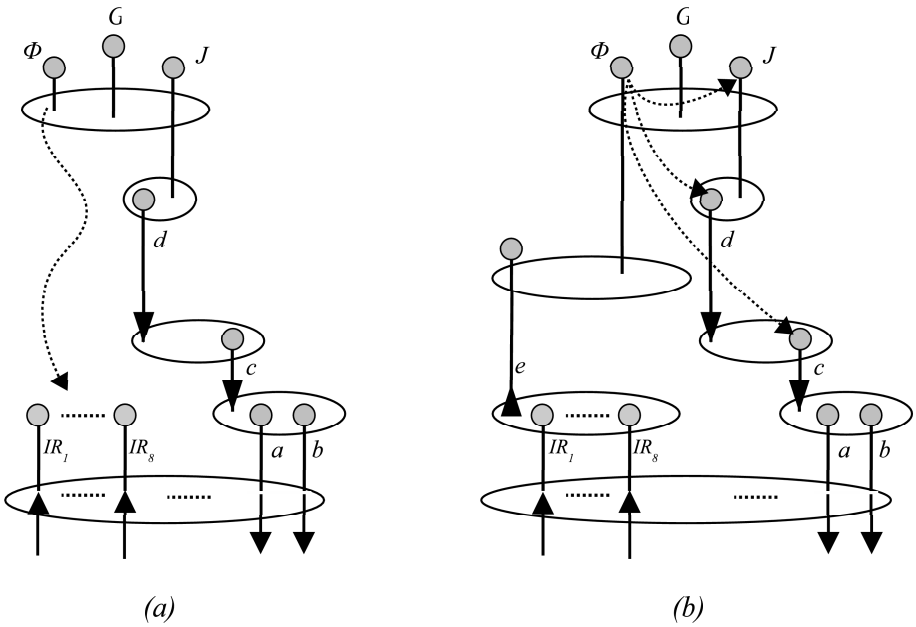

FIG. 5.7 – Recherche des causes de la chute de l'émotion Em_{Φ}. (a) Φ recherche la perception associée à la douleur. (b) Création du représentant e associé à la douleur, et recherche d'une action en cause dans la chute de l'émotion.

Puis le délégué construit une règle qui décrit la perception P_{post} sous la forme d'une conjonction. Pour l'exemple présenté en figure 5.7(a), la règle correspondant sera de la forme : $(IR_1 = \text{valeur}_1) \wedge (...) \wedge (IR_8 = \text{valeur}_8)$.

Si il n'existe pas de représentant qui regroupe déjà tous les représentants encore sélectionnés, on crée cet agent qui sera par la suite désigné par le terme R_{post}. Dans l'exemple présenté en figure 5.7(b) cela correspond à l'agent e qui a été créé par le délégué à l'intégrité physique, noté Φ. Cet agent se voit alors attribuer la règle qui vient d'être créée de sorte qu'il reconnaisse deux états : « Règle vraie » et « sinon ».

Si il existe déjà un représentant R_i qui regroupe déjà tous les représentants sélectionnés, on ajoute à celui-ci la règle qui vient d'être créée de sorte qu'il reconnaisse cette perception comme un nouvel état. Par la suite cet agent sera désigné par le terme R_{post}. Pour suivre l'exemple présenté en figure 5.7(b), l'agent e va par la suite se voir ajouter

d'autres règles de déclenchement pour des perceptions « similaires » dans le sens où elles feront intervenir les mêmes agents IR_1 à IR_8 qu'il représente déjà.

Pour déterminer A_{curr}, on va procéder de manière très similaire. On sélectionne tous les représentants moteurs pour lesquels l'écart type dans VA_{curr} est égal à zéro. On élimine de cette sélection les agents qui sont représentés par un agent pour lequel l'écart type est aussi de 0. L'agent qui reste sélectionné sera par la suite désigné par le terme R_{curr}. Dans l'exemple sur lequel nous nous appuyons dans cette sous-section, l'agent R_{curr} correspond à l'agent d. Si on observe la figure 5.8(a) on peut remarquer que celui-ci a été rajouté au groupe dans lequel se trouvent déjà le délégué à l'émotion « intégrité physique » étiquetté Φ et le représentant R_{post} précédemment créé et étiquetté e.

Si à la fin de l'exécution de ce mode de fonctionnement le délégué a trouvé un représentant capteur R_{post} ou un représentant moteur R_{curr} (ou les deux), le délégué passe dans le troisième mode. Si le délégué n'a trouvé aucun R_{post} ou R_{curr}, il retourne dans le premier mode.

Recherche de P_{prev}

La recherche de la perception P_{prev} qui précède la chute d'émotion correspond au troisième mode de fonctionnement. Pour mener à bien cette recherche le délégué doit à chaque pas de temps observer les valeurs de chaque représentant capteur. Pour cela il dispose d'un tableau permettant de retenir, pour chaque représentant capteur, la dernière valeur proposée. Ce tableau va permettre, lorsque l'émotion chute, de mettre à jour VP_{prev}, un tableau qui contient pour chaque représentant capteur un couple (valeur moyenne, écart type).

Comme pour la recherche de P_{post}, après un nombre suffisant de mises à jour du tableau VP_{prev} ce dernier est analysé par le délégué. Et de la même manière est créé un représentant R_{prev} qui réagira à la perception qui précède la chute de l'émotion. Dans l'exemple que nous exposons, l'agent R_{prev} est représenté par l'agent f de la figure 5.8(b).

Construction d'un représentant destiné à éviter la chute d'émotion

Nous sommes présentement à la fin d'exécution du troisième mode de fonctionnement aucours duquel le délégué va utiliser les représentants R_{curr} et R_{prev}. Résumons : lorsque l'on perçoit P_{prev} et que R_{curr} agit cela produit la chute de l'émotion. Il faut donc que lorsqu'on perçoit P_{prev} on inhibe l'action A_{curr} car celle-ci amènera à une insatisfaction de l'émotion.

La suite logique est donc de créer un nouveau représentant R_{anticip} qui aura pour rôle de recevoir les demandes d'action provenant de R_{curr}. Lorsque R_{anticip} percevra P_{prev}

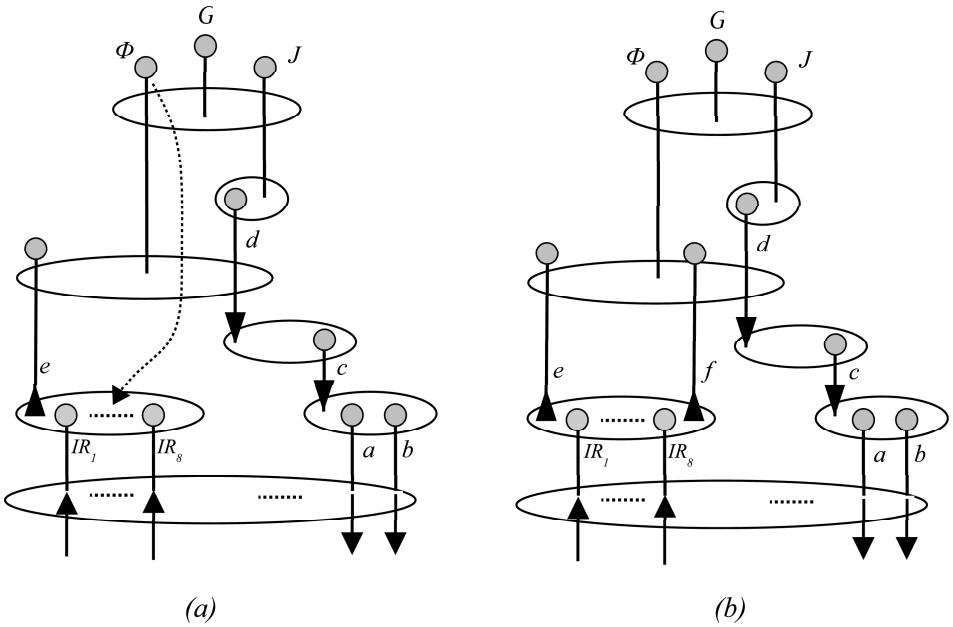

FIG. 5.8 – Recherche des causes de la chute de l'émotion Em_Φ, suite. (a) Recherche de la perception qui précède l'insatisfaction due à l'action de l'agent d. (b) Création du représentant f qui détecte la perception précédant l'insatisfaction.

il pourra alors choisir de transmettre une autre action. Cela revient à inhiber l'action A_{curr} lorsque celle ci risque de provoquer la chute de l'émotion. Ce nouveau représentant R_{curr} est représenté par l'agent h sur les figures 5.9 et 5.10.

La fonction de couplage donnée à ce nouveau représentant est une matrice d'apprentissage. Les entrées sont données par la combinaison des perceptions de l'agent R_{prev} avec les ordres émises par R_{curr}. A chaque situation différente que reconnait l'agent correspond une colonne dans la matrice présentée en table 5.1. Les sorties sont les mêmes que celles de R_{curr}.

Lors de la création de cette matrice on doit insérer des poids dans celle-ci. Lorsque l'on ne perçoit pas P_{prev}, le comportement doit être identique, c'est à dire que R_{anticip} doit transmettre les demandes d'actions qu'il reçoit. C'est pourquoi les poids correspondant ont une valeur de 1. Par contre, lorsque l'on perçoit P_{prev}, R_{anticip} va devoir inhiber la demande d'action qu'il a reçu et trouver d'autres actions à exécuter. Nous avons mis pour les poids correspondant une valeur de $\frac{1}{2}$ de sorte que ces valeurs puissent être remises en cause rapidement par le processus de renforcement.

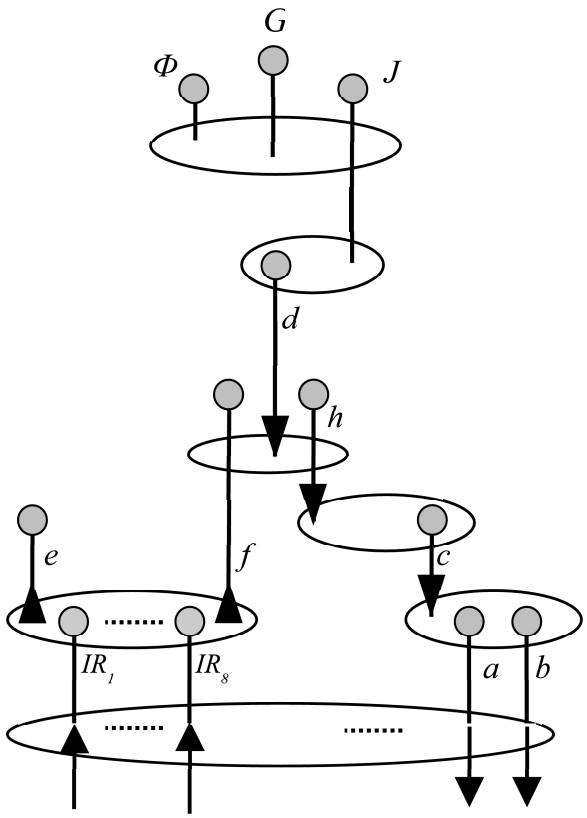

FIG. 5.9 – Création du représentant h qui, grâce à l'agent f qui perçoit les obstacles, anticipe les collisions, inhibe et remplace les ordres de d envoyés à c en conséquence. L'agent h est soumis à l'émotion Em_Φ (le lien n'est pas représenté sur le dessin). L'agent e est conservé de telle sorte qu'un autre représentant puisse l'utiliser ultérieurement.

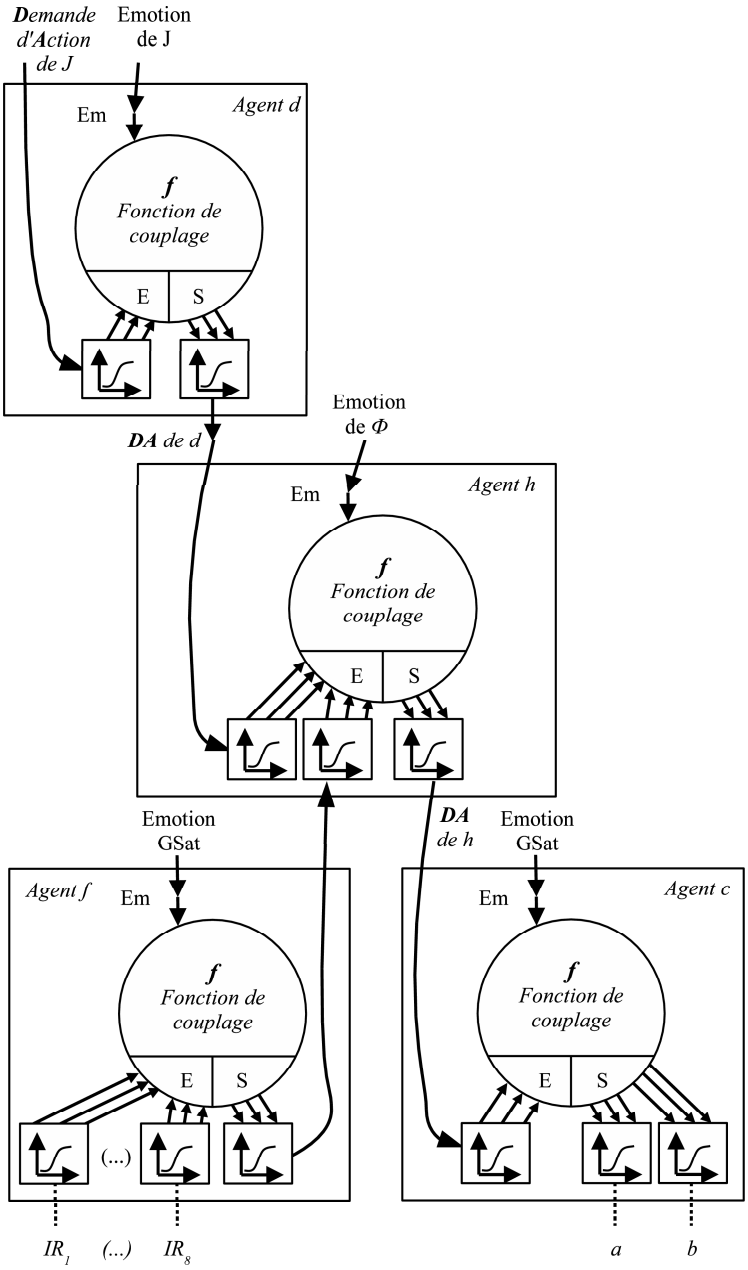

FIG. 5.10 – Structures des agents connectés au nouveau représentant h. Qrâce à l'agent f qui perçoit les obstacles le nouveau représentant h anticipe les collisions, inhibe et remplace les ordres de d envoyés à c en conséquence. L'agent h est soumis à l'émotion Em_{Φ}.

TAB. 5.1 – Poids contenus dans la matrice d'apprentissage fournie à l'agent R_{anticip} lors de sa création. Les cases vides correspondent à des poids dont la valeur est 0.

		P_{prev} perçu									Sinon								
		1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9
1	PeuPeu	½									1								
2	PeuMoy		½									1							
3	PeuBcp			½									1						
4	MoyPeu				½									1					
5	MoyMoy					½									1				
6	MoyBcp						½									1			
7	BcpPeu							½									1		
8	BcpMoy								½									1	
9	BcpBcp									½									1

5.2.4 Adaptation de la structure de contrôle

Jusqu'à présent nous avons présenté la construction de la structure de contrôle en évoquant rapidement, par endroit, le fait que certains poids contenus dans les matrices d'apprentissage allaient évoluer. Par le terme « adaptation » nous désignons le renforcement des poids contenus dans chacune des matrices. Nous allons donc compléter nos dires afin de préciser comment s'effectue cette adaptation au sein des différents agents que nous avons présenté. Il faut biensur garder en tête tous les agent disposant d'une matrice d'action vont renforcer les poids contenus dans celles-ci.

Le gestionnaire d'émotions

Le gestionnaire d'émotions est, comme nous l'avons vu, l'agent qui perçoit les émotions et qui va agir dans le but de satisfaire ces émotions. Pour un état émotionnel donné, il va choisir quelle émotion satisfaire en activant le délégué concerné. La récompense utilisée pour renforcer les actions choisies est calculée à partir de *l'état émotionnel du robot*, noté *GSat* et défini dans l'équation 4.3, sous-section 4.1.1.

De cette manière, le gestionnaire d'émotions va apprendre à donner la priorité aux délégués qui améliorent le plus l'état émotionnel du robot. Cela va permettre de donner priorité à la satisfaction des émotions qui ont l'impact le plus négatif sur l'état émotionnel du robot.

Les représentants

Chaque représentant dispose également d'une fonction de couplage qui est une matrice d'action similaire à celle présentée dans la section 4.1.2 qui traite d'apprentissage par renforcement. L'apprentissage effectué au sein de chaque représentant consiste donc à la fin d'une expérience à renforcer le poids de déclenchement de l'action qui avait été choisi en utilisant l'émotion à laquelle le représentant est connecté.

Il en résulte une optimisation locale du processus de sélection d'action qui permet donc au représentant de choisir l'action qui aura la probabilité la plus forte d'amener à la satisfaction d'une émotion. Cette optimisation locale à l'agent induit une optimisation globale de la structure de contrôle. En effet comme chaque représentant optimise son processus de sélection d'action par rapport à l'émotion à laquelle il est rattaché, le délégué qui fait directement ou indirectement appel à un tel représentant disposera de moyens plus efficaces pour satisfaire son émotion.

Si (activation par le gestionnaire d'émotions) **Alors**
  Si (Matrice d'action vide **OU** mode=recherche_sat) **Alors**
    //Rechercher un représentant
    Tester l'utilité de chaque représentant moteur du plus haut au plus bas
    Si (Il existe un représentant Rj qui peut donner satisfaction) **Alors**
      Créer un représentant Rk qui fera appel à Rj lorsque Di lui fera appel
      Ajouter l'activation de Rk à la matrice d'action de Di
    Sinon
      mode←ecoute_post
      ecoute_n←100
    Fin Si
  Sinon
    Choisir l'action au meilleur rapport (w,t)
  Fin Si
Sinon
  Si (mode=ecoute_post **ET** émotion$_i$ diminue) **Alors**
    Si (ecoute_n> 0) **Alors**
      Ecouter les perceptions reçues et les actions réalisées
      ecoute_n←ecoute_n-1
    Sinon
      Si (une perception P_{post} est associée à la chute de émotion$_i$)) **Alors**
        Créer le représentant R_{post} qui détecte P_{post}
      Fin Si
      Si (une action A_{curr} est associée à la chute de émotion$_i$)) **Alors**
        Le représentant R_{curr} est l'agent qui a déclenché l'action A_{curr}
      Fin Si
      Si (R_{post} affecté **OU** R_{curr} affecté) **Alors**
        mode←ecoute_prev
        ecoute_n←100
      Fin Si
    Fin Si
  Fin Si
  Si (mode=ecoute_prev) **Alors**
    Si (ecoute_n> 0) **Alors**
      Si (émotion$_i$ ne diminue pas) **Alors**
        PPrevTemp←capture des perceptions
      Sinon
        PPrev←PPrevTemp
        ecoute_n←ecoute_n-1
      Fin Si
    Sinon
      Si (une perception P_{prev} précède la chute de émotion$_i$)) **Alors**
        Créer le représentant R_{prev} qui détecte P_{prev}
        Créer un représentant R_{anticip} qui module les ordres de R_{curr}
        lorsque R_{prev} perçoit P_{Prev}
      Fin Si
      mode←recherche_sat
    Fin Si
  Fin Si
Fin Si

ALGORITHME 3 – Délégué à l'émotion i

Chapitre 6

Expérimentations d'apprentissage d'une structure de contrôle

Nous avons réalisé des expérimentations pour montrer que le robot développe et améliore des connaissances et des savoir-faire tout en préservant un court temps de délibération, ceci afin de montrer les propriétés du modèle : les capacités d'évolution et d'adaptation. Dans ces expérimentations, un robot mobile est placé dans un environnement dans lequel se trouvent une base de recharge, des murs et des objets qui peuvent être déplacés. Le robot perçoit trois types d'émotions comme la faim, le besoin de jouer, l'intégrité physique, et doit gérer les priorités à satisfaire chacune d'elle.

6.1 Les robots

6.1.1 Le Khepera

Afin de tester et d'évaluer les performances de notre approche, nous avons choisi d'utiliser un robot semblable au Khepera [Mondada *et al.* , 1993] présenté sur la figure 6.1. Ce robot de très petite taille est de forme cylindrique pour un diamètre de 55 mm. Comme le montre la figure 6.1, le Khepera possède suffisamment de capteurs et d'effecteurs pour lui permettre de réaliser une large gamme de tâches. Ses huit capteurs infrarouges lui permettent de détecter la proximité des objets. Il possède également deux moteurs à courant continu dont on peut choisir la vitesse de rotation et qui peuvent être pilotés indépendamment l'un de l'autre permettant au robot d'avancer, reculer et de réaliser des virages à différentes vitesses. Plusieurs modules d'extensions, tels qu'un bras manipulateur ou un système de vision, existent et peuvent être ajoutés au robot afin d'augmenter ses moyens d'interaction avec l'environnement.

Enfin, pour traiter les informations des capteurs et piloter les effecteurs, le Khepera est doté d'un processeur Motorola de la famille 68000, de 256ko de mémoire vive (RAM) et de 256ko de mémoire ROM. Quant à son autonomie d'une trentaine de minutes, elle est assurée par des batteries rechargeables NiCd.

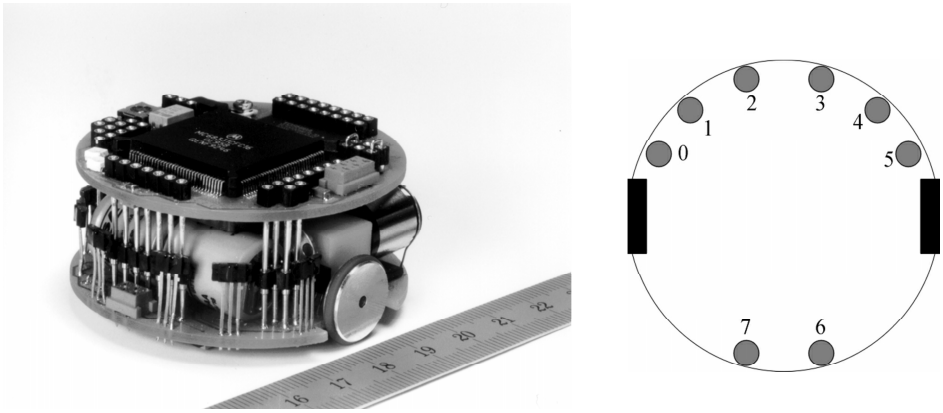

FIG. 6.1 – A gauche : photo du robot mobile Khepera. A droite : schéma du Khepera montrant la disposition de ses huit capteurs et de ses deux effecteurs.

6.1.2 Le Type 1

Dans l'optique de réaliser des expériences sur des systèmes hétérogènes, les robots *Type 1* [Lucidarme & Simonin, 2006] ont été construits au sein du Lirmm, dans le cadre du projet CoRoM [CoRoM, n.d.]. Ces robots, présentés sur la figure 6.2, ont notamment servi pour la validation des travaux d'Olivier Simonin portant sur l'étude de l'architecture Satisfaction / Altruisme [Simonin, 2001].

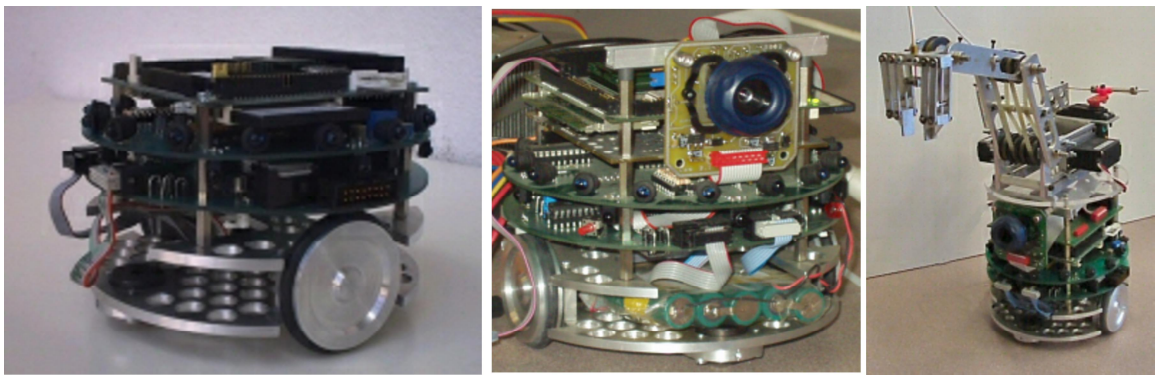

FIG. 6.2 – De gauche à droite : Premier prototype du robot mobile *Type 1*. Version améliorée du *Type 1* doté d'une caméra. Le *M3* (Miniature Mobile Manipulator) équipé d'un bras manipulateur.

Le *Type 1* est donc un robot mobile autonome équipé de deux roues différentielles montées sur des moteurs à courant continu. Il est de forme cylindrique, d'environ 13 centimètres de diamètre pour 12 centimètres de hauteur.

Capteurs de proximité

Le robot dispose d'une ceinture de capteurs de proximité constituée de 8 capteurs et de 16 émetteurs infrarouges qui lui permettent de détecter les obstacles, mais aussi de communiquer localement. Comme le montre la figure 6.3, ces 8 capteurs sont uniformément répartis sur la périphérie du robot et ont une portée de détection d'obstacles de plusieurs mètres.

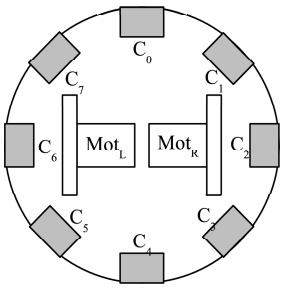

FIG. 6.3 – Disposition des capteurs et des actionneurs.

Positionnement

Le robot est également équipé d'un codeur incrémental magnétique pour chaque moteur. Ces capteurs, qui envoient 352 impulsions par tour de roue, sont utilisés pour l'odométrie : c'est-à-dire la mesure de la distance parcourue par le véhicule. Les codeurs incrémentaux permettent donc de calculer la position du robot, mais avec une dérive assez importante.

C'est pourquoi nous avons choisi un système de localisation basé sur l'effet Doppler qui permet de donner la direction et la distance d'une source émettant un signal radio. Le principe de l'effet Doppler est illustré par la figure 6.4. C'est sur ce même principe que s'appuie le système de localisation ARGOS ou encore certains calculs permettant de connaître la vitesse de déplacement d'astres célestes. La figure 6.5 montre une antenne fixe permettant de s'affranchir de l'utilisation d'une antenne mobile en simulant la rotation d'antenne. Enfin la figure 6.6 montre les résultats obtenus avec une antenne « artisanale ».

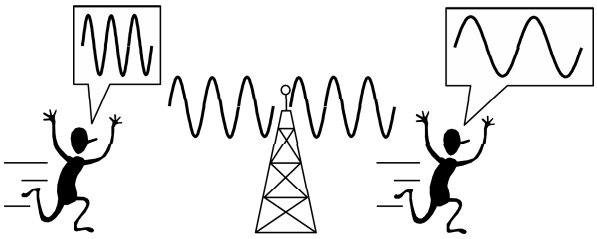

FIG. 6.4 – Illustration de l'effet Doppler. L'observateur de gauche se rapproche de la source : il perçoit une onde de fréquence plus grande que celle émise par la source. L'observateur de droite, qui s'éloigne de la source, perçoit une onde de fréquence plus petite que celle émise par la source.

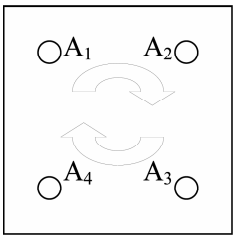

FIG. 6.5 – Antenne à effet Doppler : pour simuler le déplacement de l'observateur (l'antenne), on utilise plusieurs antennes (ici quatre : A_1 à A_4) sur lesquelles on bascule à tour de rôle. La distorsion du signal engendrée par le déplacement de l'observateur permet de déduire la direction de la source du signal.

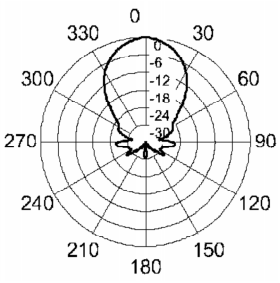

FIG. 6.6 – Cette figure montre la force du signal obtenu en fonction de l'angle. C'est la force du signal qui permet de déterminer la distance de l'objet qui émet le signal. Cette figure a été obtenue avec du matériel amateur et une antenne du même type que celle présentée sur la figure 6.5, mais qui permet tout de même une précision de l'ordre d'une dizaine de degrés.

Locomotion

Le déplacement du robot est assuré par deux roues montées sur des moteurs à courant continu. On peut donc faire varier la vitesse de rotation des moteurs et ainsi réaliser des virages plus ou moins serrés, et à différentes vitesses. La figure 6.7 montre les différentes trajectoires que l'on peut obtenir.

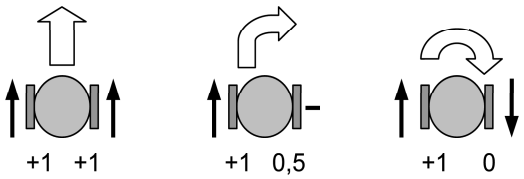

FIG. 6.7 – Trajectoire du robot en fonction des valeurs envoyées aux moteurs. A gauche : la même force est appliquée aux moteurs : le robot a une trajectoire droite. Au centre : le moteur gauche tourne plus vite que le moteur droit, le robot effectue un virage à droite. A droite : les moteurs tournent dans deux sens opposés, le robot tourne sur lui-même.

Autonomie

L'**autonomie énergétique** du robot est assurée par une paire d'accumulateurs NiMH de 1300mAH, le premier pour alimenter la partie électronique, et le second pour la partie puissance (moteurs et émission infrarouge). La partie électronique a été volontairement séparée de la partie puissance afin de ne pas être perturbée par les pics de consommation des moteurs. L'autonomie du robot varie alors entre 45 et 90 minutes en fonction de l'expérience.

L'**autonomie décisionnelle** est quant à elle assurée par un PC embarqué sur le robot. Le processeur présent sur la carte, un 486DX2 cadencé à 66MHz, lui confère une relativement grande puissance de calcul. Les 8 Mo de RAM et les 64 Mo de disque dur Compact Flash permettent la réalisation de programmes relativement complexes et le stockage d'informations de débuggage par exemple. Enfin, le BUS ISA au format PC104 permet d'empiler des cartes d'extension par-dessus le PC afin d'étendre le nombre d'entrées / sorties et ainsi accroître les capacités et les fonctionnalités du robot.

6.1.3 Comparatif Khepera / Type 1

Le tableau 6.1 récapitule les caractéristiques techniques et les capacités respectives des deux robots mobiles présentés précédemment.

Notre choix s'est porté sur le robot *Type 1* principalement en raison de son autonomie énergétique qui est au moins trois fois plus élevée que celle du *Khepera* et qui permet de

TAB. 6.1 – Comparatif Khepera / Type 1.

	Khepera	Type 1
Taille	$\phi = 5{,}5$ cm	$\phi = 13$ cm
Perception	8 capteurs infrarouges	8 capteurs infrarouges
Positionnement	Relatif (Odométrie)	Relatif (Odométrie) Absolu (repère : base)
Locomotion	2 moteurs à courant continu	2 moteurs à courant continu
Autonomie énergétique	∼10 minutes	∼60 minutes
Autonomie décisionnelle	μc Motorola 68000 à 25 MHz	μp Intel 486DX2 à 66 MHz
Mémoire vive Mémoire de masse	256ko ROM 256 ko	8 Mo Compact Flash 64 Mo
Evolutivité	Ajout de modules spécifiques	Bus ISA générique

ce fait des expérimentations de plus longue durée. De plus, l'architecture matérielle de celui-ci est un PC disposant d'un bus ISA, ce qui représente de nombreux avantages :

- l'architecture basée sur un 486DX2 66MHz autorise l'utilisation d'un système d'exploitation ne nécessitant pas de connaissances particulières en assembleur ou électronique programmable. En effet, on peut choisir le système d'exploitation à installer sur ce PC parmi les systèmes suivants : Linux, Windows 95 ou DOS. Pour des raisons d'occupation mémoire et de robustesse aux extinctions brutales, le choix s'est porté sur un DOS 6.0. Pour garantir la réactivité du système, et même si l'aspect temps réel n'est pas garanti avec le DOS, on a quand même l'assurance que le programme en cours d'exécution est le seul à pouvoir accéder au processeur, le DOS n'étant pas un système multitâche.
- l'utilisation du DOS permet de choisir parmi plusieurs compilateurs connus tels que le Quick Basic, le Turbo Pascal et le C/C++. Pour des raisons de portabilité des programmes vers d'autres plates-formes, le choix s'est porté sur l'utilisation du compilateur C/C++ distribué gratuitement par la société Borland.
- le bus ISA rend plus aisés la conception et le développement de modules additionnels comme la caméra ou le bras manipulateur présentés sur la figure 6.2.

Nous avons également choisi le *Type 1* car celui-ci est développé au sein de notre laboratoire. Il est ainsi plus aisé pour nous d'obtenir des informations sur ses spécifications techniques dans l'optique de la réalisation d'un simulateur multi-robots.

6.2 Le simulateur

L'intérêt principal de cette architecture est d'être applicable à des robots réels. Cependant, utiliser de vrais robots pour tester notre modèle d'apprentissage présente plusieurs problèmes :

- le **coût** : les robots coûtent cher et réaliser des expérimentations avec un grand nombre de robots pose des problèmes en terme de budget.
- l'**autonomie** : d'une part, les batteries des robots nécessite beaucoup de temps pour se recharger, ce qui interrompt les expérimentations pendant de longs moments. D'autre part, une autonomie de 90 minutes peut s'avérer insuffisante pour l'apprentissage de tâches complexes nécessitant des expérimentations de longue durée.
- l'**observabilité** : l'expérimentation sur des robots réels rend difficile l'observation et le débuggage du processus d'apprentissage. L'utilisation d'un simulateur rend plus aisée l'observation et les interventions sur les processus d'apprentissage.

C'est pour ces raisons que nous avons choisi de simuler des robots existants grâce à un simulateur multi-robots [Simonin *et al.* , 2002] qui s'exécute sur la plate-forme multi-agents MadKit [MadKit, 1998] [Gutknecht *et al.* , 2000].

La structure et les caractéristiques mécaniques des robots sont les mêmes que celles des robots développés dans le cadre du projet [CoRoM, n.d.], déjà utilisés dans d'autres travaux [Simonin *et al.* , 2000] [Simonin, 2001] [Lucidarme *et al.* , 2002]. Dans le simulateur, chaque capteur et effecteur est simulé avec précision en tenant compte d'un bruit aléatoire, appliqué aux perceptions, ce qui est très important pour obtenir des simulations réalistes et crédibles pour des roboticiens.

De plus, l'utilisation du simulateur nous permet, pendant la simulation, d'observer les valeurs des capteurs et de modifier manuellement les valeurs des effecteurs et des émotions. Enfin, grâce aux outils de distribution de MadKit, le simulateur peut être distribué sous forme d'applet de manière à pouvoir être diffusé sur Internet (consultable à l'adresse http ://www.lirmm.fr/~chapelle).

6.3 Une expérimentation

6.3.1 Cadre de l'expérimentation

Dans cette section, nous allons présenter un exemple d'application de notre architecture sur le robot « Type 1 » décrit dans la section 6.1.2. L'accumulation de savoir-faire se fait par exercices de difficulté croissante. Sur les figures 6.8 et 6.9 nous présentons

un exemple dans lequel un robot est équipé des huit capteurs et émetteurs infrarouges à sa périphérie lui permettant ainsi de détecter les obstacles sur le principe du radar.

Ce robot dispose également de capteurs de contact pour permettre de détecter s'il touche un objet, une collision ayant alors un impact négatif sur l'émotion qui représente l'intégrité physique. La base, grâce à laquelle le robot peut recharger ses batteries, émet un signal radio que le robot perçoit sous la forme de deux valeurs : distance et direction. Quand le robot est situé à proximité de la base, c'est-à-dire dans la zone délimitée par l'arc de cercle sur la figure 6.8, ses batteries se rechargent et il ressent un sentiment de satiété (satisfaction de la faim). Il perçoit également une émotion de jeu, qui évolue proportionnellement à la vitesse de déplacement du robot.

Enfin, un moteur factice (noté MotFact1) est également simulé. Celui-ci n'a aucun impact sur l'environnement ou sur les perceptions du robot. Il n'est présent que dans le but de montrer que le robot, remarquant son inutilité, ne l'associera pas aux autres moteurs.

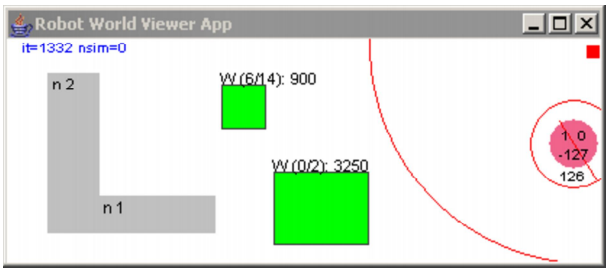

FIG. 6.8 – Capture d'écran d'une simulation en cours.

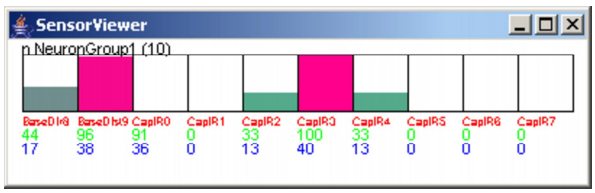

FIG. 6.9 – Les valeurs des capteurs au même moment que pour la figure 6.8 : on peut voir la réaction des capteurs infrarouges à la détection du mur.

Dans les sections qui suivent nous allons présenter une expérimentation où seront détaillées les deux phases d'apprentissage : le *babbling learning*, en section 6.3.2, puis le *satisfying learning*, en section 6.3.3.

6.3.2 Bébé babille ...

L'expérimentation débute avec un robot dont le cerveau est « vierge ». La première phase d'apprentissage est donc la phase de *babbling learning*. Pour cela le robot peut-être

placé dans un premier environnement peu complexe présenté en figure 6.10. On peut remarquer que cet environnement est vide de tout obstacle et que la zone qui permet au robot de se recharger recouvre une grande partie de l'environnement. De ce fait le robot n'aura pas, pour le moment, à se préoccuper des obstacles ou encore du niveau de charge de ses batteries. Ainsi le robot n'aura pas à gérer les émotions d'intégrité physique et de faim, respectivement liés aux collisions avec les obstacles et au niveau de charge des batteries, mais se focalisera plutôt sur la découverte de ses capacités de déplacement.

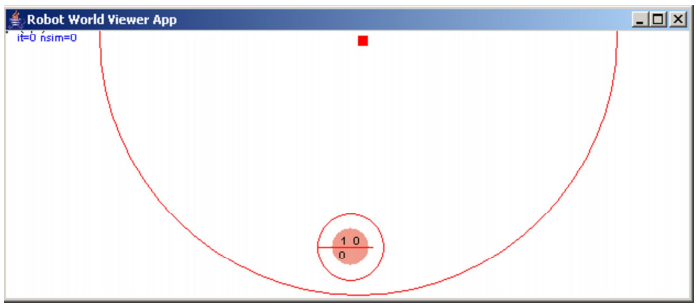

FIG. 6.10 – Environnement dans lequel le robot est placé en début d'espérience. On peut remarquer que cet environnement est vide d'obstacle et qu'il est couvert en grande partie par la zone qui permet au robot de se recharger (représentée par le grand arc de cercle).

Les figures ci-après montrent le robot actionnant ses effecteurs, présentés en haut de chaque figure, et observe ses capteurs représentés sur le bas des figures. La figure 6.11 montre le babbling sur le moteur droit, la figure 6.12 sur le moteur gauche et la figure 6.13 sur le moteur factice.

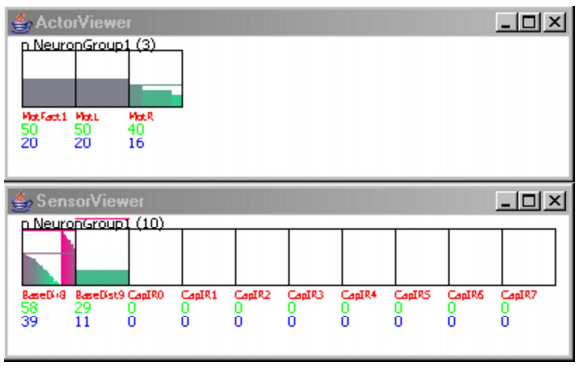

FIG. 6.11 – Impact, sur les capteurs, du moteur droit (dont la valeur est affichée à droite du vumètre des effecteurs). On remarque que le capteur de direction de la base, tout à gauche du vumètre des capteurs, change souvent de valeur.

Parmi ceux-ci, il remarque que les deux moteurs des roues ont un impact sur le capteur qui donne la direction de la base et sur celui qui représente la distance à la base

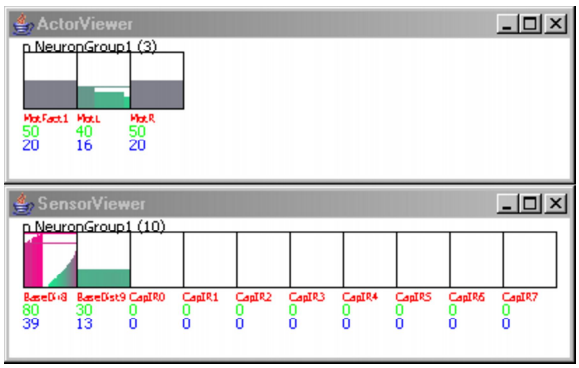

FIG. 6.12 – Impact du moteur gauche sur le capteur de direction de la base.

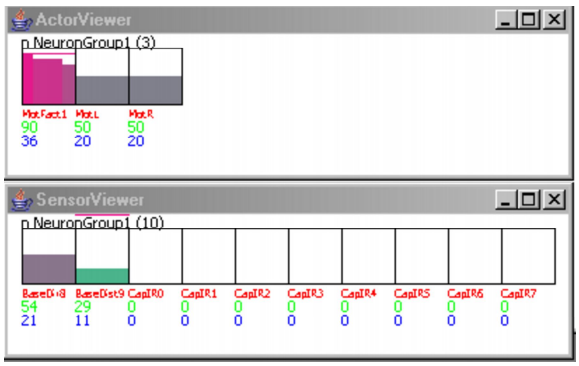

FIG. 6.13 – Impact du moteur factice sur les capteurs. On note que la valeur du capteur de direction de la base n'évolue pas, et cela est normal puisque le robot est immobile.

(chacun des moteurs pouvant tourner dans un sens ou dans l'autre). Dans le listing 6.2, la direction de la base est l'avant-dernière composante du vecteur d'impact (la valeur la plus élevée), et la distance à la base est la dernière valeur.

Sur ce listing apparaît également le coefficient de corrélation pour chaque moteur (ici MotR et MotFact1) par rapport au premier moteur testé (MotL). Ainsi l'impact sur les capteurs engendré par le moteur MotR est plus proche de celui de MotL, que MotFact1. En effet, le coefficient de corrélation entre MotL et MotR est d'environ 0.814, alors que entre MotL et MotFact1, il n'est que de 0.0010.

TAB. 6.2 – Listing de debug montrant les tests de similarité.

```
Babbler: initialisé (1)
Babbler: Fin de test premier moteur (MotL)
SensorsImpactV1= impact de(MotL):
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 21.08 0.35
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.01 0.01
Babbler: Fin de test d'un des OtherMotors (MotR)
SensorsImpactV2= impact de(MotR):
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 17.15 0.28
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.01 0.01
analyseImpact:angle(VA,VB)=0.0 // Distance=3.924
analyseImpact:sigma_angle=0.999 // sigma_distance=0.814 // sigma=0.814
Motors :MotL(V1=21.08/2630) and MotR(V2=17.15/2630)
Moteur MotR sera ajouté au groupe (sigma=0.814)
Babbler: Fin de test d'un des OtherMotors (MotFact1)
SensorsImpactV2= impact de(MotFact1):
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.03 0.01
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.01 0.01
analyseImpact:angle(VA,VB)=0.071 // Distance=21.049
analyseImpact:sigma_angle=0.977 // sigma_distance=0.0010 // sigma=0.001
Motors :MotL(V1=21.08/2630) and MotFact1(V2=0.03/2630)
Moteur MotFact1 ne sera PAS ajouté au groupe (sigma=0.0010)
```

Il crée donc un groupe contenant les deux agents moteurs, ainsi qu'un représentant qui sera responsable de proposer les différentes combinaisons motrices intéressantes, parmi lesquelles : avancer, reculer, tourner à gauche, à droite, faire un demi tour sur lui même, ou ne pas bouger. Le listing 6.3 détaille le déroulement de la fin de cette phase de babbling : création du représentant, du groupe associé et activation de celui-ci par le *neuron scheduler*.

TAB. 6.3 – Listing de debug montrant la fin du babbling et la création du représentant des moteurs.

```
Babbler: Babbling terminé, MotorGroup contient:
–MotL–MotR–
Nsch:babbling terminé
Agent neurone initialisé ...
(NeuronGroup1 Neurone2 smaapp.NeuronScheduler@1b19753)
Activation NeuralAgent en cours...
NeuronGroup1...ok ...
group NeuronGroup1
Neuron activated: Neurone2
(NSched):smaapp.NeuronAgent@1e6978d registered as Neuron
Nsch:neurone MotLMotR activé
```

6.3.3 Ses premiers pas ...

Une fois la phase de babbling terminée, le gestionnaire d'émotions va essayer des actions parmi celles disponibles et observer l'impact sur les émotions. Les actions vont être choisies parmi celles proposées par les délégués aux émotions, puis par les représentants de groupe d'effecteurs, comme le représentant du déplacement créé durant la phase de babbling, et enfin par les actions possibles sur tous les effecteurs. Le gestionnaire d'émotions commence donc par essayer les représentants déjà existants. Le délégué au jeu remarque alors que parmi les actions proposées par le représentant du déplacement, beaucoup satisfont le jeu, à l'exception de l'action « ne pas bouger ». Un groupe est alors créé par le délégué au jeu, et un nouveau représentant y est placé afin de proposer les actions connues pour satisfaire la faim. Ce représentant, noté *J1* sur la figure 6.15 ci-dessous, est donc une solution au problème de jeu.

Ce premier savoir-faire acquis, on augmente la difficulté du problème et on place le robot dans un environnement contenant des obstacles. Pour assouvir son besoin de jeu, le robot effectue des déplacements et entre alors en collision avec un mur. Le délégué à l'intégrité physique perçoit de ce fait une chute de l'émotion qu'il représente. Il cherche donc parmi les actions possibles celles qui vont résoudre son problème, et trouve dans les neuf actions proposées par le représentant du déplacement, un sous-ensemble d'actions qui permet au robot de s'éloigner de l'obstacle. Le délégué à l'intégrité physique crée alors un groupe dans lequel un nouveau représentant est placé afin de pouvoir résoudre ce problème par la suite.

Le robot se remet à jouer et rencontre d'autres murs. Le gestionnaire d'émotions remarque que quand le robot joue, l'intégrité physique peut chuter indépendamment

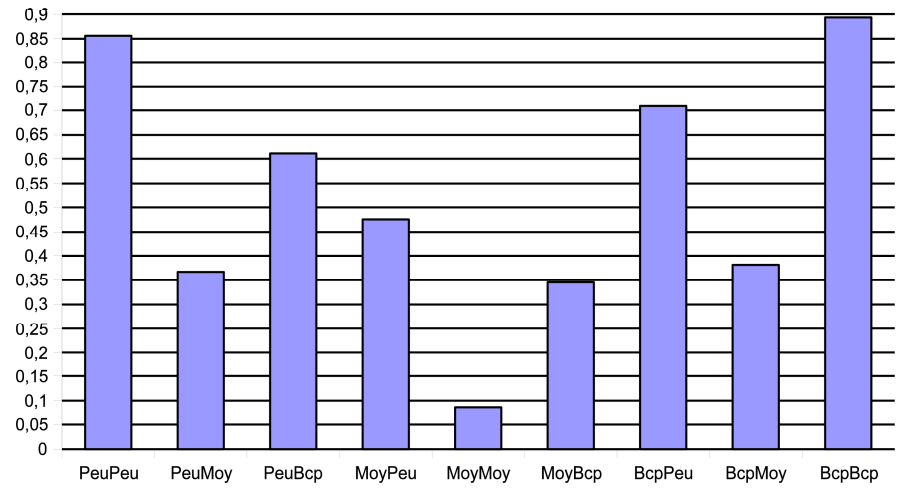

FIG. 6.14 – Histogramme des poids de déclenchement.

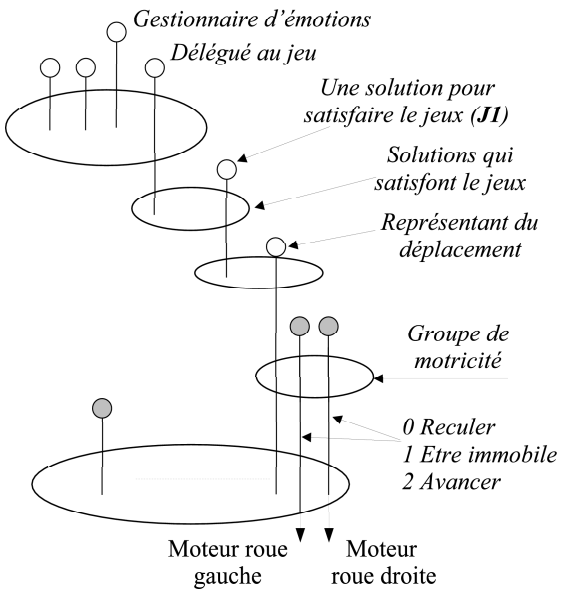

FIG. 6.15 – Structure organisationnelle résultant de l'apprentissage d'un comportement qui satisfait le jeu.

TAB. 6.4 – Matrice d'apprentissage.

Action		Roue gauche	Roue droite	Poids / Nb essais	
1	PeuPeu	Reculer	Reculer	0,856	79
2	PeuMoy	Reculer	Immobile	0,367	80
3	PeuBcp	Reculer	Avancer	0,611	62
4	MoyPeu	Immobile	Reculer	0,476	84
5	MoyMoy	Immobile	Immobile	0,087	64
6	MoyBcp	Immobile	Avancer	0,345	81
7	BcpPeu	Avancer	Reculer	0,709	64
8	BcpMoy	Avancer	Immobile	0,382	80
9	BcpBcp	Avancer	Avancer	0,893	84

de l'action entreprise précédemment. Il cherche donc parmi les perceptions celles qui interviennent à chaque fois qu'il rencontre ce problème.

Il remarque que la chute d'intégrité physique coïncide avec la valeur « beaucoup » du capteur infrarouge situé devant lui. Cette perception représente donc une exception à prendre en compte dans la procédure de jeu *J1* apprise précédemment. Le gestionnaire de jeu crée donc un nouveau représentant *J2*, illustré sur la figure 6.16. Ce dernier fait appel à *J1* sauf quand un obstacle est détecté. Dans ce cas, *J2*, va apprendre à sélectionner, parmi les autres actions possibles, celle qui permet d'éviter, et donc d'anticiper, la collision.

6.3.4 Une tâche plus complexe

6.4 Discussion

Dans les premières expérimentations que nous avons réalisées, nous avons observé l'émergence de certains comportements connus comme l'évitement d'obstacle ou la recherche de la base de recharge quand le besoin se fait ressentir.

Au fur et à mesure, nous avons complexifié le problème en enrichissant l'environnement d'objets déplaçables et en ajoutant une autre émotion gratifiant le déplacement de ces objets dans une zone de dépôt. La structure de contrôle obtenue après de nombreux cycles d'apprentissage exprime alors plusieurs comportements satisfaisant à tour de rôle chacune des émotions ressenties par le robot.

Ces résultats nous encouragent à continuer le développement de notre simulateur afin de réaliser plus d'expérimentations dans des environnements variés et sur des robots d'architecture différente. De plus, il serait intéressant de doter le simulateur de la

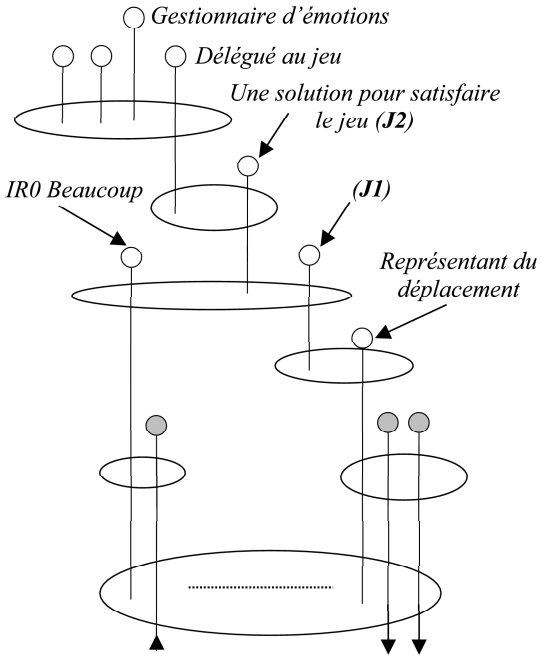

FIG. 6.16 – La structure de la figure 6.15 s'est enrichie de *J2* pour anticiper les collisions.

capacité à sauvegarder la structure de contrôle du robot pour permettre de la présenter sous une forme compréhensible par l'utilisateur.

Chapitre 7

Conclusion

7.1 Bilan

Nous avions défini dans le premier chapitre la problématique qui nous intéressait, à savoir de proposer une architecture de contrôle générique qui permette de s'adapter à l'environnement et à la structure physique du robot sur laquelle elle est déployée, et qui émerge par apprentissage en fonction des besoins primaires qu'éprouve le robot.

Cette thèse a donc été construite afin de proposer une réponse à cette problématique sous la forme d'un modèle d'architecture de contrôle et d'une implémentation montrant sa faisabilité. Nous reprenons ici le cheminement qui nous a amené à un tel modèle et précisons nos différentes contributions.

Notre travail a d'abord consisté à observer ce qui existait déjà en termes d'architectures de contrôle apprenantes. Nous avons donc commencé par observer ce qui se faisait dans la nature en terme de système nerveux, et avons étudié différents modèles existants de ces systèmes biologiques. Les réseaux de neurones artificiels présentant certaines limitations pour permettre l'apprentissage de comportements et de tâches complexes, nous nous sommes orientés sur une modélisation du cortex en unités fonctionnelles de haut niveau : les colonnes et les aires corticales.

Nous nous sommes alors retrouvés confrontés à une différence fondamentale entre des architectures massivement parallèles dans les systèmes biologiques et des architectures matérielles dont nous disposons pour mettre en place une implémentation de ces modèles biologiques. Il a donc été nécessaire d'étudier les architectures de contrôle existantes en Intelligence Artificielle afin de déterminer quel compromis nous souhaitions faire entre des modèles biologiques et des modèles faisant intervenir d'imposants modèles mathématiques et logiques. Nous avons alors été plus particulièrement attiré par des architectures distribuées mettant en jeu un ensemble d'entités apprenantes réactives.

Puis nous nous sommes demandé comment devait s'opérer l'apprentissage d'une telle structure et pour quelles raisons un robot pouvait avoir besoin de déclencher cet apprentissage. Nous avons alors mis en évidence la nécessité de modéliser les différents besoins d'un robot afin de motiver et de guider l'apprentissage vers des comportements destinés à satisfaire ces besoins.

Nous avons donc proposé un concept d'*émotion* afin de modéliser les différents besoins d'un robot et nous sommes appuyés sur ce concept pour proposer un modèle d'architecture de contrôle à base d'agents capable d'apprendre des tâches complexes et de s'adapter à des évolutions de l'environnement ou de sa structure physique. C'est pourquoi nous avons défini plusieurs types d'agents permettant de réaliser une structure de contrôle dont les comportements servent à satisfaire les émotions perçues par le robot. Puis nous nous sommes appuyés sur cette modélisation pour proposer des mécanismes d'apprentissage permettant de conférer à la structure de contrôle des propriétés d'adaptation et d'évolution. A cet effet, nous avons défini deux phases d'apprentissage susceptibles de créer la structure de contrôle et les différents agents qui la composent, dans le but d'obtenir une architecture aux comportements satisfaisants.

L'apport essentiel de ce travail est donc d'avoir défini un modèle d'architecture de contrôle s'appuyant sur des concepts déjà développés séparément en intelligence artificielle et qui rassemblés permettent d'obtenir une architecture évolutive et adaptative. Les points importants de notre contribution sont :

- L'utilisation d'un *Système Multi-Agents* pour modéliser la structure de contrôle du robot et les diverses entités autonomes qui la constituent
- La réalisation d'une architecture *émotionnelle* afin de guider l'apprentissage du robot sur les comportements lui permettant de se satisfaire
- L'intégration d'un algorithme d'*apprentissage par renforcement* pour permettre l'adaptation des agents qui constituent l'architecture
- La définition d'une phase préliminaire d'apprentissage par *babillage* capable de faire émerger de la structure physique du robot un ensemble de structures de contrôle proches du schéma sensori-moteur du robot
- La définition d'une seconde phase d'apprentissage réalisant la construction de structures destinées à produire des comportements satisfaisant les émotions, et permettant donc de répondre aux divers besoins du robot

Enfin, ce modèle a été implémenté sur un simulateur que nous avons développé dans le but de reproduire de manière réaliste les contraintes liées au robot que nous avions choisi pour les expérimentations.

Nous avons alors obtenu des résultats intéressants en terme de comportements satisfaisant les émotions, mais aussi certains comportements a priori absurdes mais qui se

sont révélés a posteriori parfaitement cohérents avec les contraintes de satisfaction des besoins que nous avions données à nos robots.

7.2 Perspectives

Lorsque nous avons effectué nos travaux de recherche et d'implémentation, nous avons dû nous limiter sur certains points qui mériteraient d'être développés afin d'accélérer le processus d'apprentissage ou de permettre la réalisation de comportements encore plus complexes. Ces travaux ouvrent également une voie dans l'étude de comportements animaux et humains en ce qui concerne la validation de modèles appartenant aux domaines de l'éthologie et de la psychologie.

Apprentissage des émotions

Les méthodes d'apprentissage que nous avons exposées ont permis la construction d'une structure de contrôle venant s'intercaler entre la couche émotionnelle et la couche physique. Nous avons vu que le gestionnaire d'émotions avait alors la possibilité de choisir quelle émotion satisfaire en se basant sur un algorithme d'apprentissage par renforcement dont les récompenses était données par la moyenne, éventuellement pondérée, des émotions.

Mais il serait intéressant de permettre à ce gestionnaire d'émotions de construire une structure de contrôle venant s'intercaler entre lui et les délégués aux émotions. De cette manière on obtiendrait une architecture qui, en plus d'apprendre des comportements satisfaisants, apprendrait également à gérer ses émotions. Cela pourrait permettre de mieux comprendre des comportements humains liés à des troubles émotionnels.

Gestion du temps

Le modèle que nous avons proposé ne prend pas le temps en compte, que ce soit dans le processus de sélection d'action ou dans le processus d'apprentissage. L'architecture de contrôle est effectivement basée sur les capteurs et les effecteurs dont dispose le robot. Une première idée serait d'ajouter à la couche physique un capteur supplémentaire délivrant, par exemple, une impulsion à chaque pas de temps, ou une valeur continue qui augmenterait au fur et à mesure que le temps s'écoule. Cependant, si l'on procède ainsi nous pensons que le robot devra dépenser beaucoup d'efforts d'apprentissage, et construire de nombreux représentants avant de commencer à pouvoir utiliser la notion de temps.

Il existe donc un réel travail de recherche pour pouvoir intégrer le concept de temps à notre modèle.

Psychologie artificielle

Ce modèle s'inspirant de modèles biologiques tels que la modélisation corticale et les émotions, il serait intéressant d'effectuer une étude sur les capacités de ce modèle à faire émerger des comportements a priori incohérents liés à des expériences traumatisantes.

Lors d'expérimentations que nous avons menées nous avions observé ce type de trouble du comportement. En effet, lors d'une expérimentation le moteur physique du simulateur s'était arrêté, mais le cerveau du robot avait continué à être actif. Après avoir réactivé le moteur physique qui simulait l'environnement du robot nous avions découvert un robot « traumatisé ». Celui-ci avait passé une longue période pendant laquelle toutes les actions qu'il faisait ne produisaient aucun changement de l'environnement. De ce fait tous les comportements qu'il avait appris avaient été remis en cause par une égalisation des poids dans les matrices d'action. Le résultat était un robot qui hésitait sans cesse entre les différents comportements qu'il avait appris avant son « traumatisme ».

D'autres erreurs d'implémentation nous ont également permis d'observer des modifications comportementales inattendues. En effet, lors d'une expérimentation, une partie des valeurs qui devaient transiter d'un représentant à un autre n'arrivaient jamais à destination. On peut comparer cet effet à celui de produits psychoactifs qui bloqueraient en partie les communications de signaux entre les neurones d'un système nerveux biologique. Nous avons d'ailleurs observé les mêmes effets sur le comportement du robot que ceux de produits psychoactifs : le robot réagissait parfois avec un temps de retard, ce qui produisait des trajectoires aléatoires amenant quelquefois le robot à rencontrer un mur.

Nous pensons donc qu'il serait intéressant d'étudier grâce à ce modèle les impacts sur le comportement induits par une expérience traumatisante ou par l'influence de produits sur les transmissions d'informations. Pour cela, on pourrait traumatiser volontairement des robots disposant déjà d'une structure de contrôle ou encore simuler les effets de divers produits psychoactifs. Cela permettrait de mieux comprendre des troubles du comportement observés chez les espèces vivantes.

Plus de fonctions de couplage

Dans notre modèle, nous avons défini les représentants comme des entités disposant d'une fonction de couplage f mettant en relation les entrées et les sorties par rapport à une émotion. Puis nous avons proposé pour réaliser cette fonction de couplage d'utiliser des matrices d'actions telles que celles utilisées en apprentissage par renforcement. Cela

nous a déjà permis d'obtenir des comportements intéressants. Néanmoins cette méthode est mal adaptée à des problèmes de vision ou de reconnaissance de la voix. Pour cela, nous avons vu que les réseaux de neurones donnaient de bons résultats.

Il serait donc intéressant de voir comment on pourrait intégrer cette méthode d'apprentissage à notre modèle afin de permettre à notre architecture de contrôle de disposer plus rapidement de capacités perceptives comme la vision ou la perception de sons et de mots. De manière générale il serait intéressant de donner la possibilité au robot d'utiliser différentes fonctions de couplage ainsi que les moyens de détecter les situations qui nécessitent l'utilisation de telles fonctions.

Des architectures matérielles distribuées

Nous avons rapidement évoqué le fait que les architectures matérielles de calcul évoluaient vers des architectures à base de processeurs multi-cœurs par opposition aux architectures actuelles assez monolithiques.

Cela ouvre des perspectives quant à l'utilisation même de systèmes multi-agents pour bénéficier de la puissance offerte par les milliers de cœurs que l'on trouvera sur les processeurs qui sortiront d'ici une dizaine d'années.

De telles architectures ouvrent également des perspectives quant à la réalisation d'architectures de contrôle distribuées comme celle que nous avons proposée. En effet de telles architectures matérielles devraient permettre de concevoir des architectures de contrôle dont on pourrait augmenter les capacités cognitives par l'ajout, au besoin, de processeurs multi-cœur supplémentaires.

Annexe A

Similarité d'impact sur les capteurs

A.1 Similarité d'impact sur les capteurs de deux moteurs

Lors de la phase de babbling, les effecteurs ayant le même impact sur les capteurs sont regroupés entre eux. Ce regroupement prend également en compte l'impact émotionnel des actions sur l'agent.

Dans cette sous-section nous allons donc détailler les outils mathématiques qui permettent de définir l'impact d'un moteur sur les n capteurs comme un vecteur à n dimensions. Enfin, nous nous appuyons sur les calculs de distance et d'angle pour définir le coefficient de similarité qui nous permet de grouper deux moteurs ayant un fonctionnement similaire.

A.2 Impact d'un moteur sur les capteurs

L'impact d'un moteur m_k sur les n capteurs c_j est un vecteur de dimension n où chaque composante j représente la variation de la valeur v du capteur c_j pendant que l'on stimule le moteur m_i. Ainsi, la variation d'un capteur c_j à l'instant t est donnée par :

$$\Delta_t^{c_j} = \sum_{i=0}^{t} |v_i - v_{i-1}| * |\mathrm{GSat}_i - \mathrm{GSat}_{i-1}| \tag{A.1}$$

On peut noter que dans cette équation intervient le calcul de la variation de l'état émotionnel du robot, noté GSat. Cela permet de donner plus d'importance aux variations associées à des changements importants de l'état émotionnel du robot.

A partir de là, on définit donc le vecteur $\overrightarrow{\mathrm{I}_t^{m_k}}$ qui représente l'impact d'un moteur m_k sur l'ensemble des n capteurs à l'instant t :

$$\overrightarrow{\mathrm{I}_t^{m_k}} = \begin{pmatrix} \Delta_t^{c_1} \\ \Delta_t^{c_2} \\ \vdots \\ \Delta_t^{c_n} \end{pmatrix} \tag{A.2}$$

A.3 Distance

Dans un plan (espace à deux dimensions), la distance entre deux points $A(x_1, y_1)$ et $B(x_2, y_2)$:

$$\text{distance(A,B)} = \sqrt{(x_2 - x_1)^2 + (x_2 - x_1)^2} \tag{A.3}$$

Lorsque l'on manipule des vecteurs dans un espace à n dimensions, la distance entre deux vecteurs $\vec{\mathrm{V}_1}$ et $\vec{\mathrm{V}_2}$ est donnée par :

$$\text{distance}(\vec{\mathrm{V}_1}, \vec{\mathrm{V}_2}) = \sqrt{\sum_{i=0}^{n} (V_2[i] - V_1[i])^2} \tag{A.4}$$

A.4 Angle

Pour calculer l'angle formé par deux vecteurs dans $\mathbb{R}^n$, il faut utiliser le **produit scalaire** et le **produit vectoriel**. Et pour le calcul de ces deux produits, nous devons d'abord définir la norme d'un vecteur. Le calcul de celle-ci découle du calcul de distance décrit dans l'équation A.4. En effet, la norme d'un vecteur $\vec{V}$ exprimé sur n dimensions, est la distance entre l'origine et l'extrémité du vecteur. La norme est donc définie par :

$$\left\| \vec{V} \right\| = \sqrt{\sum_{i=0}^{n} V[i]^2} \tag{A.5}$$

L'angle formé par les directions des deux vecteurs $\vec{\mathrm{V}_1}$ et $\vec{\mathrm{V}_2}$ est donné par les calculs du produit scalaire et du produit vectoriel.

A.4.1 Calcul du produit scalaire

Le **produit scalaire** de deux vecteurs, noté $\vec{\mathrm{V}_1} \cdot \vec{\mathrm{V}_2}$, est le nombre :

$$\vec{\mathrm{V}_1} \cdot \vec{\mathrm{V}_2} = \left\| \vec{\mathrm{V}_1} \right\| \cdot \left\| \vec{\mathrm{V}_2} \right\| \cdot cos\theta \quad \text{avec} \quad 0 \leq \theta \leq \pi \tag{A.6}$$

θ peut donc être déduit de :

$$arcos\left(\frac{\vec{V_1}\cdot\vec{V_2}}{\left\|\vec{V_1}\right\|\cdot\left\|\vec{V_2}\right\|}\right)=\left\{\begin{array}{c}\theta+2\cdot k\cdot\pi\\ \text{ou}\\ -\theta+2\cdot k\cdot\pi\end{array}\right. \quad \text{avec} \quad 0\leq\theta\leq\pi \tag{A.7}$$

A.4.2 Calcul du produit vectoriel

Le **produit vectoriel** de deux vecteurs $\vec{V_1}$ et $\vec{V_2}$ définis dans $\mathbb{R}^n$, noté $\vec{V_1}\wedge\vec{V_2}$, est le vecteur $\vec{V_3}$ défini par :

$$\vec{V_3}=\vec{V_1}\wedge\vec{V_2}=\left\|\vec{V_1}\right\|\cdot\left\|\vec{V_2}\right\|\cdot sin\theta\cdot\vec{u} \quad \text{avec} \quad 0\leq\theta\leq\pi \tag{A.8}$$

Les n composantes du vecteur $\vec{V_3}$ peuvent être calculées par :

$$\vec{V_1}\begin{pmatrix}V_1[0]\\ \vdots\\ V_1[i]\\ \vdots\\ V_1[n-1]\end{pmatrix}\wedge\vec{V_2}\begin{pmatrix}V_2[0]\\ \vdots\\ V_2[i]\\ \vdots\\ V_2[n-1]\end{pmatrix}=$$

$$\vec{V_3}\begin{pmatrix}V_3[0]=(V_1[1]\cdot V_2[2])-(V_1[2]\cdot V_2[1])\\ \vdots\\ V_3[i]=(V_1[(i+1)\%n]\cdot V_2[(i+2)\%n])-(V_1[(i+2)\%n]\cdot V_2[(i+1)\%n])\\ \vdots\\ V_3[n-1]=(V_1[0]\cdot V_2[1])-(V_1[1]\cdot V_2[0])\end{pmatrix} \tag{A.9}$$

A.4.3 Calcul de l'angle

Dans l'équation A.8 $\vec{u}$ est un vecteur unitaire défini dans $\mathbb{R}^{n+1}$ et orthogonal à l'espace $\mathbb{R}^n$ dans lequel $\vec{V_1}$ et $\vec{V_2}$ sont définis, et de telle façon que $(\vec{V_1},\vec{V_2},\vec{u})$ forment un système orienté. On peut donc réécrire A.8 :

$$\left\|\vec{V_1}\wedge\vec{V_2}\right\|=\left\|\vec{V_1}\right\|\cdot\left\|\vec{V_2}\right\|\cdot sin\theta\cdot\|\vec{u}\| \tag{A.10}$$

Puisque $\vec{u}$ est unitaire (c'est-à-dire de longueur 1), $\|\vec{u}\| = 1$, l'équation A.10 peut se réécrire :

$$arcsin\left(\frac{\left\|\vec{V_1} \wedge \vec{V_2}\right\|}{\left\|\vec{V_1}\right\| \cdot \left\|\vec{V_2}\right\|}\right) = \begin{cases} \theta + 2 \cdot k \cdot \pi \\ \text{ou} \\ \pi \cdot \theta + 2 \cdot k \cdot \pi \end{cases} \tag{A.11}$$

Or, la fonction $arccos\,(\alpha)$ est définie de l'intervalle $[-1, 1]$ à l'intervalle $[0, \pi]$ (équation A.12, illustrée par la figure A.1), et la fonction $arcsin\,(\alpha)$ est définie de l'intervalle $[-1, 1]$ à l'intervalle $[\frac{-\pi}{2}, \frac{\pi}{2}]$ (équation A.13, illustrée par la figure A.2).

$$arccos : [-1, 1] \rightarrow [0, \pi] \tag{A.12}$$

$$arcsin : [-1, 1] \rightarrow [-\frac{\pi}{2}, \frac{\pi}{2}] \tag{A.13}$$

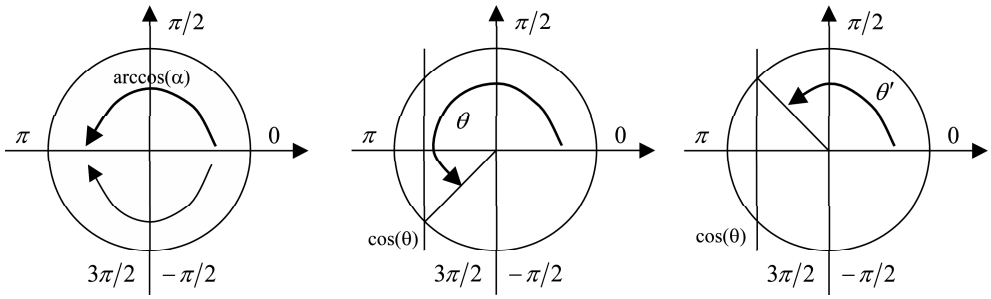

FIG. A.1 – A gauche : l'$arccos(\alpha)$ est exprimé entre 0 et π. Au centre : l'angle θ qui a le même cosinus que l'angle θ' de la figure de droite et qui est renvoyé par $arccos(\alpha)$.

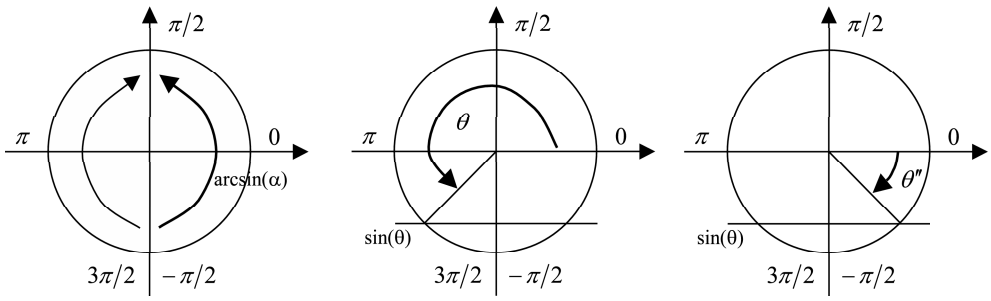

FIG. A.2 – A gauche : l'$arcsin(\alpha)$ est exprimé entre $-\frac{\pi}{2}$ et $\frac{\pi}{2}$. Au centre : l'angle θ qui a le même sinus que l'angle θ'' de la figure de droite et qui est renvoyé par $arcsin(\alpha)$.

Pour obtenir l'angle θ, il faut donc prendre en compte la valeur de $arccos\,(\alpha)$ et de $arcsin\,(\alpha)$:

$$\text{si} \quad arcsin\left(\frac{\left\|\vec{V}_1 \wedge \vec{V}_2\right\|}{\left\|\vec{V}_1\right\| \cdot \left\|\vec{V}_2\right\|}\right) > 0 \quad \begin{array}{ll} \text{alors} & \theta = arcos\left(\frac{\vec{V}_1 \cdot \vec{V}_2}{\|\vec{V}_1\| \cdot \|\vec{V}_2\|}\right) \\ \text{sinon} & \theta = 2 \cdot \pi \cdot -arcos\left(\frac{\vec{V}_1 \cdot \vec{V}_2}{\|\vec{V}_1\| \cdot \|\vec{V}_2\|}\right) \end{array} \tag{A.14}$$

A.5 Coefficient de similarité

Pour déterminer si l'impact $\vec{V_k}$ du moteur m_k est proche de l'impact $\vec{V_1}$ du premier moteur testé m_1, il faut que :

- la distance entre $\vec{V_1}$ et $\vec{V_k}$ soit faible,
- l'angle formé par les deux vecteurs soit faible.

C'est pourquoi le calcul du coefficient de similarité σ prend en compte le coefficient de similarité des distances σ_d et d'angle σ_θ :

$$\sigma = \sigma_d \cdot \sigma_\theta \tag{A.15}$$

Le coefficient de similarité des distances est le nombre $\sigma_d \in [0, 1]$ donné par le ratio entre la plus petite norme et la plus grande :

$$\sigma_d = \frac{min(\left\|\vec{V}_1\right\|, \left\|\vec{V}_2\right\|)}{max(\left\|\vec{V}_1\right\|, \left\|\vec{V}_2\right\|)} \tag{A.16}$$

Pour le calcul du coefficient de similarité d'angle $\sigma_\theta \in [0, 1]$, il faut prendre en considération que si l'angle formé par les deux vecteurs est proche de π, cela signifie que les deux vecteurs sont radicalement opposés. Par contre lorsque l'angle vaut $2 \cdot \pi$, les vecteurs sont parallèles. Le coefficient de similarité d'angle est donc inversement proportionnel à la distance de l'angle θ à π. Ramené à l'intervalle $[0, 1]$, cela donne :

$$\sigma_\theta = \begin{cases} 1 - \frac{\theta}{\pi} & \forall \theta \leq \pi \\ 1 - \frac{2 \cdot \pi - \theta}{\pi} & \forall \theta > \pi \end{cases} \tag{A.17}$$

Références bibliographiques

[Alexandre, 1990]Alexandre, Frédéric. 1990 (Avril). *Une modelisation fonctionnelle du cortex : la colonne corticale, aspects visuels et moteurs.* Thèse de Doctorat, Université Henry Pointcaré.

[Alexandre, 1997]Alexandre, Frédéric. 1997. *Intelligence neuromimetique.* Habilitation à Diriger des Recherches, Université Henry Pointcaré, Nancy I, France.

[Beaugé & Alexandre, 1995]Beaugé, Lionel, & Alexandre, Frédéric. 1995. *Neuromodulation Mechanisms for the Cooperation of Artificial Neural Networks.* Tech. rept. CRIN–CNRS / INRIA-Lorraine.

[Boutilier, 1999]Boutilier, Craig. 1999. Sequential Optimality and Coordination in Multiagent Systems. *Pages 478–485 of : Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99.* Morgan Kaufmann.

[Brooks, 1986]Brooks, Rodney A. 1986 (March). A Robust Layered Control System for a Mobile Robot. *Pages 14–23 of : IEEE Journal of Robotics and Automation*, vol. 2.

[Burnod, 1989]Burnod, Yves. 1989. *An adaptative neural network : the cerebral cortex.* Masson.

[Chapelle *et al.* , 2002]Chapelle, Jerome, Simonin, Olivier, & Ferber, Jacques. 2002. How Situated Agents can Learn to Cooperate by Monitoring their Neighbors' Satisfaction. *Pages 68–72 of :* van Harmelen, Frank (ed), *Proceedings of the 15th Eureopean Conference on Artificia Intelligence, ECAI'2002.* IOS Press.

[Chapelle, 2001]Chapelle, Jérôme. 2001 (4 Juillet). *Apprentissage de la coopération dans les systèmes multi-agents situés hétérogènes.* Rapport de DEA, Université Montpellier II.

[Connell, 1990]Connell, Jonathan H. 1990. *Minimalist mobile robotics : a colony-style architecture for an artificial creature.* San Diego, CA, USA : Academic Press Professional, Inc.

[CoRoM, n.d.]CoRoM. ***Co****opération et coordination de* ***Ro****bots* ***Mo****biles (Cooperation and Coordination of Multi-robot System), http ://www.lirmm.fr/w3rob/.*

[Cortex, n.d.]Cortex. *Intelligence neuromimétique, http ://cortex.loria.fr/.*

[Cullogh & Pitts, 1943]Cullogh, Warren S. Mac, & Pitts, Walter. 1943. A logical calculus of the ideas immanent in nervous activity. *Pages 115–133 of : Bulletin of Mathematical Biophysics 5.*

[Damasio, 1994]Damasio, Antonio R. 1994. *Descartes' Error : Emotion, Reason and the Human Brain.*

[Drogoul, 1993]Drogoul, Alexis. 1993 (23 Novembre). *De La Simulation Multi-Agent A La Résolution Collective de Problèmes : Une Étude De l'Émergence De Structures D'Organisation Dans Les Systèmes Multi-Agents.* Thèse de Doctorat, Université Paris VI.

[Dutech *et al.* , 2001]Dutech, Alain, Buffet, Olivier, & Charpillet, François. 2001. Multi-Agent Systems by Incremental Gradient Reinforcement Learning. *Pages 833–838 of : Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001.* Morgan Kaufmann.

[Edelman, 1992]Edelman, Gerald M. 1992. *Bright Air, Brilliant Fire : On the Matter of the Mind.* Basic Books.

[Ferber, 1995]Ferber, Jacques. 1995. *Les Systèmes Multi-Agents : Vers une intelligence collective.* InterEditions, Paris.

[Ferber, 1999]Ferber, Jacques. 1999. *Multi-Agent Systems : An Introduction to Distributed Artificial Intelligence.* Addison-Wesley Longman Publishing Co., Inc.

[Foliot & Michel, 1998]Foliot, Gérald, & Michel, Olivier. 1998. Learning object significance with an emotion based process. *Pages 25–30 of : In SAB'98 Workshop on Grounding Emotions in Adaptive Systems.*

[Frankowska *et al.* , 2003]Frankowska, H., Burnod, Y., & Dufossé, M. 2003. *Modeling the Satisficing-Learning in Basal Ganglia : a Viability Approach.*

[Frezza-Buet, 1999]Frezza-Buet, Hervé. 1999 (27 octobre). *Un modele de cortex pour le comportement motive d'un agent neuromimetique autonome.* Thèse de Doctorat, Université Henry Pointcaré.

[Frolov *et al.* , 2002]Frolov, Alexander A., Dufossé, Michel, Bensmail, Soraya, & Ouezdou, Fethi Ben. 2002. Biologically Inspired Neural Network Approach to Manipulator Movement Control. *Pages 3876–3881 of : Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA 2002.* IEEE.

[Gadanho & Hallam, 2001a]Gadanho, Sandra Clara, & Hallam, John. 2001a. Emotion–triggered learning for autonomous robot control. *Pages 531–559 of : Cybernetics and Systems – Special Issue : Grounding emotions in adaptative systems*, vol. 32. D. Cañamero, C. Numaoka, P. Petta.

[Gadanho & Hallam, 2001b]Gadanho, Sandra Clara, & Hallam, John. 2001b. Robot learning driven by emotions. *Adaptative Behavior*, **9**(1), 42–64.

[Gutknecht *et al.* , 2000]Gutknecht, Olivier, Ferber, Jacques, & Michel, Fabien. 2000. Une expérience d'architecture de plateforme multi-agent générique. *Page 224 of :* Hermès

(ed), *Journées Francophones pour l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents JFIADSMA'00.*

[Hebb, 1949]Hebb, D. O. 1949. *The Organization of Behavior.* John Wiley, New York.

[Hopfield, 1982]Hopfield, J.J. 1982. Neural networks and physical systems with emergent collective computational abilities. *Pages 460–464 of : Proceedings of the National Academy of Sciences.*

[Ito, 1984]Ito, Masao. 1984. *The Cerebellum and neural control.* Raven Press, New York.

[James L. McClelland & the PDP Research Group, 1986]James L. McClelland, David E. Rumelhart, & the PDP Research Group. 1986. *PARALLEL DISTRIBUTED PROCESSING Explorations in the Microstructure of Cognition Volume 1 et 2.* Cambridge, Massachusetts : The MIT Press.

[Kohonen, 1990]Kohonen, Teuvo. 1990 (September). The self-organizing map. *Pages 1460–1480 of : Proceedings of IEEE*, vol. 78.

[Lucidarme, 2003]Lucidarme, Philippe. 2003. *Apprentissage et adaptation pour des ensembles de robots réactifs coopérants.* Thèse de Doctorat, LIRMM, Université Montpellier II.

[Lucidarme & Simonin, 2006]Lucidarme, Philippe, & Simonin, Olivier. 2006. Le robot mobile Type 1. *In : Journée des démonstrateurs en robotique.*

[Lucidarme *et al.* , 2002]Lucidarme, Philippe, Simonin, Olivier, & Liégeois, Alain. 2002 (11–15 may). Implementation and Evaluation of a Satisfaction/Altruism Based Architecture for Multi-Robot Systems. *Pages 1007–1012 of : Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA 2002.*

[MadKit, 1998]MadKit. 1998. *MadKit, a **M**ulti–**A**gent **D**evelopment **Kit**, site officiel : http ://www.madkit.org.*

[Maes, 1991]Maes, Pattie. 1991. A bottom-up mechanism for behavior selection in an artificial creature. *Pages 238–246 of : Proceedings of the First International Conference on Simulation of Adaptive Behavior.* Cambridge, MA, USA : MIT Press.

[Maçãs *et al.* , 2001]Maçãs, Márcia, Ventura, Rodrigo, Custódio, Luis, & Pinto-Ferreira, Carlos. 2001. Experiments with an emotion-based agent using the DARE architecture. *Pages 105–112 of : Proceedings of the Symposium on Emotion, Cognition, and Affective Computing, AISB'01 Convention, University of York, UK, 2001.*

[Matarić, 1992]Matarić, Maja J. 1992. Minimizing complexity in controlling a mobile robot population. *Pages 830–835 of : Proceedings of the 1992 IEEE International Conference on Robotics and Automation, ICRA 1992.* IEEE.

[Matarić, 1994a]Matarić, Maja J. 1994a. *Interaction and Intelligent Behavior.* Thèse de Doctorat, Massachusetts Institute of Technology, Cambridge, MA, USA.

[Matarić, 1994b]Matarić, Maja J. 1994b. Leaning to behave socially. *Pages 453–462 of : SAB94 : Proceedings of the third international conference on Simulation of adaptive behavior : from animals to animats 3.* Cambridge, MA, USA : MIT Press.

[Minsky, 1986]Minsky, Marvin. 1986. *The society of mind.* Simon & Schuster, Inc.

[Minsky & Papert, 1969]Minsky, Marvin, & Papert, Seymour. 1969. *Perceptrons : An Introduction to Computational Geometry.* Cambridge, MA, USA : The MIT Press.

[Mondada *et al.* , 1993]Mondada, Francesco, Franzi, Edoardo, & Ienne, Paolo. 1993. Mobile Robot Miniaturization : A Tool for Investigation in Control Algorithms. *Pages 501–513 of :* Yoshikawa, Tsuneo, & Miyazaki, Fumio (eds), *Proceedings of the Third International Symposium on Experimental Robotics, Kyoto, Japan.* Springer Verlag.

[Parker, 1998]Parker, Lynne E. 1998 (April). ALLIANCE : An architecture for fault-tolerant multi-robot cooperation. *Pages 220–240 of : IEEE Transactions on Robotics and Automation*, vol. 14(2).

[Revel, 1997]Revel, Arnaud. 1997 (Novembre). *Contrôle d'un robot autonome par approche neuro-mimétique.* Thèse de Doctorat, ENSEA.

[Rosenblatt, 1958]Rosenblatt, F. 1958. The Perceptron : A probabilistic model for information storage and organization in the brain. *Psychological Review*, p. 386–408.

[Shibata *et al.* , 1996]Shibata, T., Ohkawa, K., & Tanie, K. 1996. Spontaneous behavior of robots for cooperation–Emotionally intelligent robot system. *Pages 2426–2531 of :* Mathieu, Philippe, & Müller, Jean-Pierre (eds), *In Proceedings of IEEE International Conference on Robotics and Automation.* Hermès, Paris.

[Simonin, 2001]Simonin, Olivier. 2001 (20 décembre). *Le modèle satisfaction-altruisme : coopération et résolution de conflits entre agents situés réactifs, application à la robotique.* Thèse de Doctorat, Université Montpellier II.

[Simonin & Ferber, 2000]Simonin, Olivier, & Ferber, Jacques. 2000 (4–6 october). Modeling Self Satisfaction and Altruism to handle Action Selection and Reactive Cooperation. *Pages 314–323 of : SAB 2000, The Sixth International Conference on the Simulation of Adaptative Behavior.*

[Simonin *et al.* , 2000]Simonin, Olivier, Liégeois, Alain, & Rongier, Philippe. 2000. An Architecture for Reactive Cooperation of Mobile Distributed Robots. *Pages 35–44 of :* L.E. Parker, G. Bekey, J. Barhen (ed), *DARS'00 : 5^{th} International Symposium on Distributed Autonomous Robotic Systems.* Springer.

[Simonin *et al.* , 2002]Simonin, Olivier, Michel, Fabien, Chapelle, Jérôme, & Ferber, Jacques. 2002. Un simulateur de systèmes multi-robots dans MadKit. *Pages 167–170 of :* Mathieu, Philippe, & Müller, Jean-Pierre (eds), *$X^{èmes}$ Journées Francophones pour l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents JFIADSMA'02.* Hermès, Paris.

[Sony, 2006]Sony. 2006. *AIBO global link : http ://www.sony.net/Products/aibo/.*

[Sutton & Barto, 1998]Sutton, R. S., & Barto, A. G. 1998. *Course Notes : Reinforcement Learning I : An Introduction.* MIT Press.

[Zadeh, 1965]Zadeh, Lotfi A. 1965. Fuzzy Sets. *Information and Control*, **8**, 338–358.

Liste des tables

Liste des figures

Index

Résumé

Cette thèse se situe à la rencontre du domaine des systèmes multi-agents et de la robotique. Ce manuscrit présente la structure de contrôle d'un robot autonome et l'architecture multi-agents utilisée pour la modéliser. Cette architecture permet au robot de s'adapter, d'évoluer et d'apprendre de nouvelles tâches de manière non supervisée.

Pour obtenir une structure de contrôle qui s'adapte à la structure physique du robot, à l'environnement de celui-ci, et aux tâches à exécuter, nous nous sommes inspirés des systèmes nerveux des espèces vivantes. Ainsi, nous nous inspirons des colonnes corticales pour définir des entités apprenantes qui regroupent le fonctionnement de plusieurs neurones au sein d'une seule entité. Nous modélisons l'organisation de cette structure et les interactions entre entités apprenantes en utilisant un système multi-agents. Cela nous permet de donner à cette structure de contrôle des propriétés d'adaptation et d'évolution pour répondre aux besoins du robot et des tâches qu'il doit réaliser.

Afin de guider la construction, le fonctionnement et l'évolution de cette structure nous introduisons le concept de motivations dont le principe se rapproche de celui des émotions. Certaines motivations représentent des besoins liés au bon fonctionnement du robot (comme l'intégrité physique). D'autres motivations sont liées à des objectifs que le développeur souhaite faire atteindre au robot. Ces motivations sont utilisées dans un processus d'apprentissage par renforcement pour remplacer le superviseur qui était responsable des récompenses, et permettre ainsi d'obtenir un apprentissage autonome. Nous définissons alors les deux phases d'apprentissage qui permettent de déterminer la dynamique de la structure physique du robot puis de mémoriser les objectifs intermédiaires qui découlent de la réalisation d'une tâche complexe.

La structure de contrôle proposée est évaluée par l'implémentation sur un simulateur de robots mobiles. Nous détaillons également dans ce manuscrit la structure des robots utilisés pour ces expérimentations ainsi que les résultats d'apprentissage obtenus lors des simulations.

Mots-clés : Systèmes Multi-Agents, Apprentissage, Agent émotionnel, Robotique.

A Multi-agent architecture for an emotion driven learning

Abstract

In this thesis, we present a multi-agent architecture giving a robot the ability to learn in an unsupervised way. An autonomous learning is achieved by using emotions which represent basic needs for the learning entity. The learning process we propose is inspired by the organization in cortical columns and areas of a living being brain. The organizational multi-agent architecture is used to describe the interaction among entities involved in the learning process

Keywords : Multi-Agent Systems, Machine Learning, Emotional agent, Robotics.